NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

BALLISTIC MISSILE INTERCEPTION FROM UCAV

by

Zheng Liang Lu

December 2011

Thesis Advisor:	Oleg Yakimenko
Second Reader:	Chris Brophy

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2011	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Ballistic Missile Intercept from UCAV			**5. FUNDING NUMBERS**
6. AUTHOR(S) Zheng Liang Lu			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The objective of this thesis is to conduct a study to evaluate the feasibility of the hit-to-kill trajectory-shaping(TS) guidance of an air-launched missile from a UCAV against enemy ballistic missiles via computer simulation, using a TS-guidance algorithm developed by LT Lukacs and Prof Yakimenko based on the direct method of calculus of variations that maximizes the kinetic energy transfer of an air-launched missile against an aerial target. The computer simulation code will generate the air-launched missile's entire flight path in order to minimize the distance travelled by the air-launched missile, minimize the time to intercept, and maximize kinetic energy transfer to the target (a simulated enemy missile) by controlling the interception geometry while providing near-optimal flight path to interception. This will be done by utilizing the direct method of calculus of variations combined with inverse dynamics theory to generate, in real time, an optimal flight path using the missile's onboard sensors and computers. The results have confirmed the feasibility of hit-to-kill trajectory-shaping(TS) guidance of an air-launched anti-ballistic missile from a UCAV.

14. SUBJECT TERMS Ballistic Missile Interception, UCAV, Trajectory-Shaping, Guidance Algorithm, Direct Method of Calculus			**15. NUMBER OF PAGES** 75
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

BALLISTIC MISSILE INTERCEPT FROM UCAV

Zheng Liang Lu
Assistant Principal Engineer, Singapore Technologies Aerospace
B.A.(Engineering Tripos), University of Cambridge(UK), 2004
Master of Engineering, University of Cambridge(UK), 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2011

Author: Zheng Liang Lu

Approved by: Professor Oleg Yakimenko
Thesis Advisor

Professor Christopher Brophy
Second Reader

Professor Knox Millsaps
Chair, Department of Mechanical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this thesis is to conduct a study to evaluate the feasibility of the hit-to-kill trajectory-shaping(TS) guidance of an air-launched missile from a UCAV against enemy ballistic missiles via computer simulation, using a TS-guidance algorithm developed by LT Lukacs and Prof Yakimenko based on the direct method of calculus of variations that maximizes the kinetic energy transfer of an air-launched missile against an aerial target. The computer simulation code will generate the air-launched missile's entire flight path in order to minimize the distance travelled by the air-launched missile, minimize the time to intercept, and maximize kinetic energy transfer to the target (a simulated enemy missile) by controlling the interception geometry while providing near-optimal flight path to interception. This will be done by utilizing the direct method of calculus of variations combined with inverse dynamics theory to generate, in real time, an optimal flight path using the missile's onboard sensors and computers. The results have confirmed the feasibility of hit-to-kill trajectory-shaping(TS) guidance of an air-launched anti-ballistic missile from a UCAV.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1

II. MODELING AND SIMULATION....5
- A. DESCRIPTION OF TRAJECTORY SHAPING GUIDANCE....5
- B. SIMULATION SOFTWARE ARCHITECTURE....6
  - 1. Flow-Chart....6
- C. BALLISTIC MISSILE TARGET MODELING....6
- D. UCAV-LAUNCHED INTERCEPTOR MISSILE MODELING....8
- E. MISSILE FLIGHT PROGRAM....12
  - 1. Initialization....12
  - 2. Flight Program Description....13

III. TRAJECTORY SHAPING GUIDANCE AND ITS BENEFITS....15

IV. SIMULATED LAUNCHES AND RESULTS....17
- A. SIMULATION RESULTS (LAUNCH ALT = 10KM, LAUNCH AIRSPEED=100M/S OR MACH 0.33)....17
- B. LAUNCH ALT = 17KM, LAUNCH AIRSPEED=100M/S....21

V. DISCUSSIONS, ADDITIONAL RESULTS AND ANALYSIS....25

VI. CONCLUSION....31

LIST OF REFERENCES....33

APPENDIX....35
- A. NCADEFLIGHTFROMUCAV.M....35
- B. NCADEGUIDANCE.M....40
- C. NCADETRAJECTORY.M....46
- D. NCADEPARAMS1.M....53

INITIAL DISTRIBUTION LIST....57

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Phases of Ballistic Missile Trajectory (From [8])....2
Figure 2. Simulation Flowchart....6
Figure 3. Reach of TPD-2 Missile....7
Figure 4. TPD-2 Ballistic Missile Flight Path simulated in MATLAB(From [1]) 8
Figure 5. X-47B at take-off....9
Figure 6. X-47B at cruise....9
Figure 7. X-47B Specifications (From [11])....10
Figure 8. NCADE dimensions (From [9,10])....11
Figure 9. Interceptor Stage (Stage 2) (From [9])....11
Figure 10. Interceptor Trajectory at Default Conditions (Northing: 100km, Westing: 100km, Weighing:1000) 18
Figure 11. Interceptor Euler Angles....19
Figure 12. Interceptor Trajectory at Default Conditions (Northing: 100km, Westing: 100km, Weighing:1000) 19
Figure 13. Lateral Acceleration (LATAX, Ny and Nz) Load Factor (No or negligible dynamic constraint violations 20
Figure 14. Interceptor Missile Impact Angle and Time-to-go 20
Figure 15. Composite Interceptor Trajectory from Different Directions (the individual interceptor trajectories are 3D) 21
Figure 16. Kinematic Boundary (max radius 60km) of Interceptor Missile at launch alt =17km (all constraints satisfied)....22
Figure 17. Kinematic Boundary(max radius 60km) of Interceptor Missile at launch alt =17km (additional plots)....23
Figure 18. Trajectories at a lower launch altitude of 10km compared to the original launch altitude of 17km 25
Figure 19. Interceptor Trajectories at launch altitude of 17km (90deg Impact Angle weighing and non-weighing comparison)....26
Figure 20. Boundary increased to 150km with new launch altitude of 10km and 90deg Impact Angle requirement....27
Figure 21. Kinematic Boundary increased to 150km with new launch altitude of 10km and 90deg Impact Angle requirement removed (additional plots) 28
Figure 22. Various launch altitudes compared (weighing for 90 deg impact angle removed) 29

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Characteristics(estimated) TPD-2 ICBM Data Input to Simulation(From [1])....7
Table 2. Characteristics(estimated) of NCADE Input into Simulation (From [8,9,10,11])....12

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ABM	Anti-Ballistic Missile
AMRAAM	Advanced Medium-Range Air-to-Air Missile (AIM-120)
DoF	Degrees of Freedom
CAP	Combat Air Patrol
CG	Centre of Gravity
DARPA	Defense Advanced Research Projects Agency (US)
DoD	Department of Defence (US)
EFC	Electronic Forward Closure
ICBM	Inter-Continental Ballistic Missile
J-UCAS	Joint Unmanned Combat Air Systems (US)
LOS	Line-of-Sight
NCADE	Net-Centric Airborne Defence Element (AIM-120 missile derivative for use against Ballistic Missiles)
PN	Proportional Navigation
TS	Trajectory Shaping
UCAV	Unmanned Combat Air Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author wishes to thank his wife, Serene, for her patience and understanding when he was doing the thesis.

The author would also like to thank his thesis advisor, Professor Oleg Yakimenko, for his invaluable guidance and patience throughout the academic year.

Above all, the author thanks God for everything. Soli Deo gloria.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Ballistic missiles are getting more prevalent amongst world militaries. Such missiles can carry heavy payloads deep behind enemy lines compared to conventional tube artillery, providing a very powerful weapon at relatively nominal cost. Because of their high terminal velocity, ballistic missiles are difficult to intercept. Current air defense systems have improved ability to intercept tactical ballistic missiles, but cannot protect assets against these missiles with certainty. This allows a moderate force of missiles to threaten a numerically superior enemy force by penetrating their air defenses better than with conventional aircraft, while providing a deeper strike than tube artillery. A ballistic missile is only guided during the relatively brief initial powered phase of flight, known as the boost phase, and its course is subsequently governed by the laws of orbital mechanics and ballistics.

To ensure a successful intercept of a supersonic ballistic missile, the interceptor must have the necessary response time, speed and accuracy. The interception can take place in any of the three distinct phases of ballistic missile flight—boost, midcourse or terminal (see Figure 1).

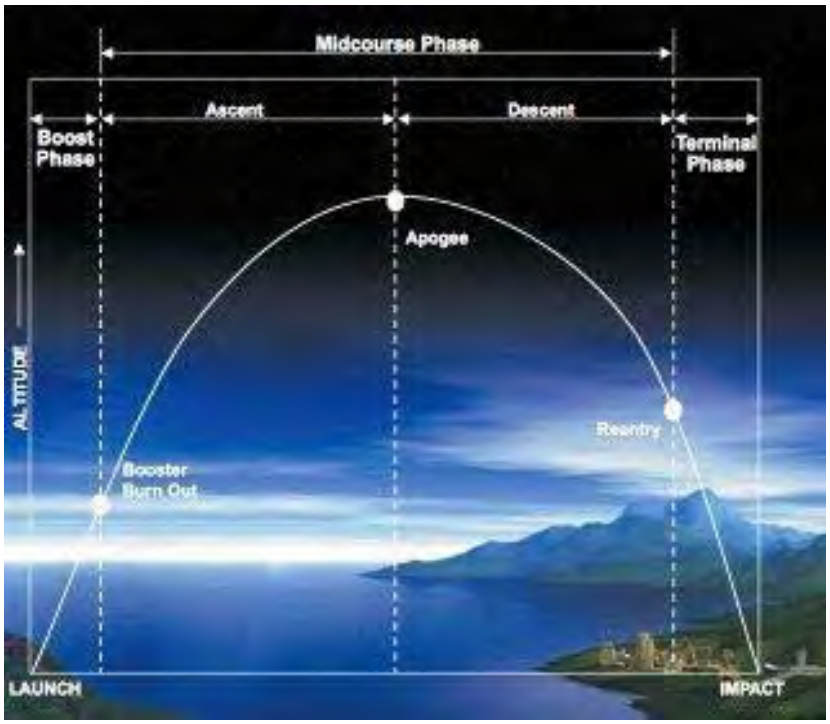

Figure 1. Phases of Ballistic Missile Trajectory (From [8])

During the boost phase, the ballistic missile rocket motors are burning to accelerate the missile to a high trajectory altitude. Intercepting a ballistic missile in its boost phase is the ideal solution for missile defence, as the ballistic missile is most vulnerable during this phase of its flight. The reasons are that the hot rocket exhaust plume makes detection and targeting easier. The disadvantage however, lies in the geographical siting of the interceptor system (which has to be close to hostile territory) and the short time to intercept, typically about 180 seconds. The trajectory-shaping(TS) guidance developed by Lukacs and Prof Yakimenko in 2006 [1] shows that boost phase interception (BPI) is possible with a high probability of certainty. The disadvantage of the need for siting the interceptor system close to hostile territory is negated by the use of air-to-air anti-ballistic missiles mounted on a stealthy X-47B UCAV conducting a Combat Air Patrol (CAP). This thesis focuses on the air-to-air interception of ballistic missiles in the boost phase.

There are different guidance laws for the missile interception of aerial target, including the various types of proportional navigation (PN) guidance. A new guidance

algorithm was developed by John A. Lukacs IV and Prof Yakimenko in 2006 to intercept a ballistic missile during the boost phase by a sea-based missile interceptor. This TS guidance algorithm uses the direct method of calculus of variations that maximizes the kinetic energy transfer from a surface-launched missile to a ballistic missile target.

A trade-off study was previously conducted [5] by applying this guidance law in simulated ballistic missile interception from a ship-launched SM-6 ABM as the interceptor missile and examined the interactions and trade-offs between the various critical parameters in the intercept solution, like the endgame intercept geometry, time-to-intercept and intercept altitude. This provided insights into the feasibility and limitations of the TS guidance algorithm. A literature review of the drag model used in the algorithm and comparison of the new guidance with the compensated PN guidance was also conducted, and an induced drag model was developed for future studies.

This project extends this to verify if an air-launched interceptor model based on the AIM-120 NCADE missile from the same manufacturer would be able to accomplish similar or superior performance with the TS guidance algorithm. The results verified that the trajectory-shaping guidance is feasible for the interception of ballistic missiles in the boost phase from a UCAV for a great range of launch locations with respect to a ballistic missile detection point by the UCAV.

THIS PAGE INTENTIONALLY LEFT BLANK

II. MODELING AND SIMULATION

A. DESCRIPTION OF TRAJECTORY SHAPING GUIDANCE

In 2006, Lukacs and Prof Yakimenko [1] developed a simulation code in MATLAB as a demonstration of the feasibility of intercepting a ballistic missile during the boost phase by a surface-launched interceptor missile using the Trajectory Shaping (TS) guidance law. This thesis seeks to build upon this to show how the TS guidance law can be used for air-launched interceptor missiles. The TS guidance uses the principle of flight path optimization from the interceptor to the predicted target position. It relies on high-order polynomial as a reference function for the flight path and uses the virtual domain in the optimization process. A preset thrust history is used in the computation of interceptor flight path. The flight path is derived by minimization a combined performance index including intercept geometry, time-to-intercept, penalty on altitude and dynamic constraints. This performance index (J) is given by the following equation,

$$J = t_{go}^2 + W_{IA}(\mu - \mu_{desired})^2 + P_1 + P_2$$

where t_{go} is the time-to-intercept, μ is the impact angle, P_1 is the penalty on intercept altitude and P_2 is the penalty on dynamic constraints. The application of such guidance is particularly suitable for interception of targets with a predictable trajectory, such as a ballistic missile in the boost phase.

In the simulation, the U.S.-made NCADE missile, was used as the interceptor , while the ballistic missile target was modelled on the DPRK TPD-2 ballistic missile. A 3 degree-of-freedom (3DoF) mathematical model was previously developed and used to simulate the trajectory and flight characteristics of both the ballistic and interceptor missile based on open-source missile data. The intercept path is continuously calculated onboard the interceptor missile as a two-point boundary value problem, using Direct Methods of Calculus of Variation to calculate a near-optimal flight path and the control commands necessary to attain it.

B. SIMULATION SOFTWARE ARCHITECTURE

1. Flow-Chart

The 3-DoF Simulation Modeling flowchart used in this project is shown. It was derived from research work previously carried out by Prof Yakimenko, Lukacs [1] and Leong [5].

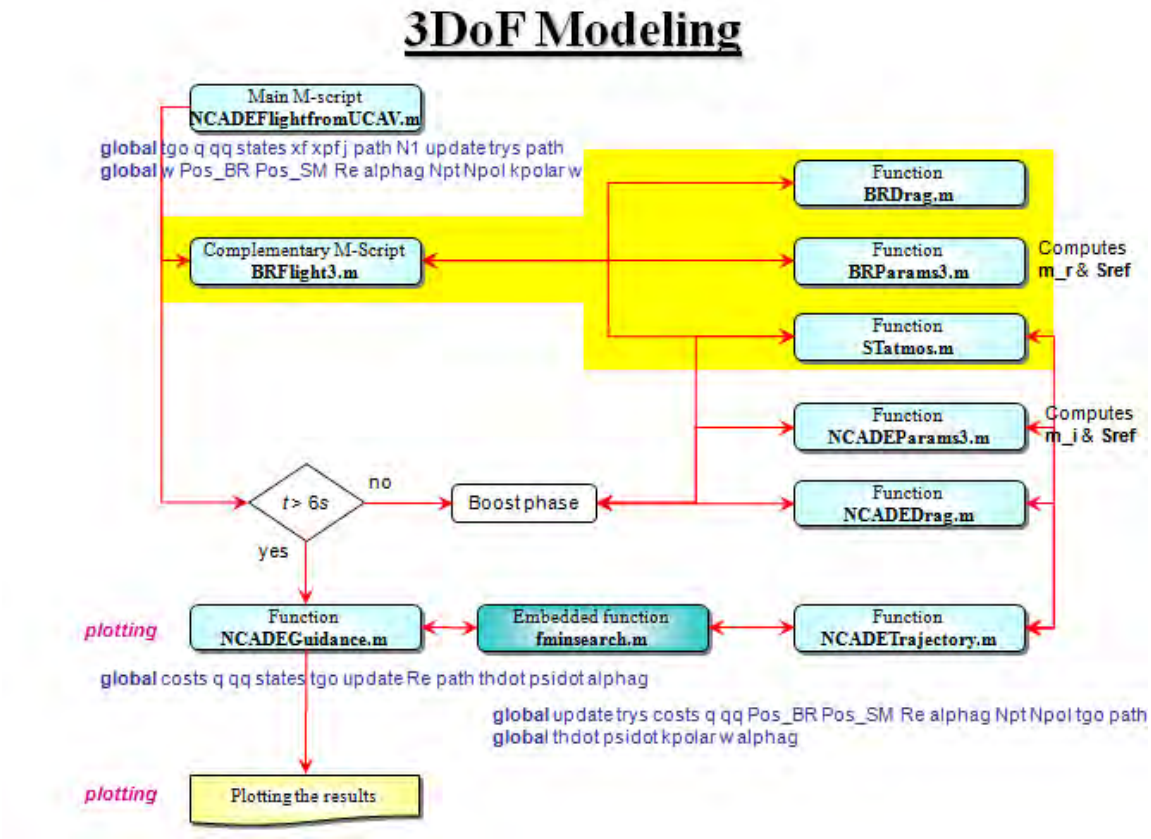

Figure 2. Simulation Flowchart

C. BALLISTIC MISSILE TARGET MODELING

The TPD-2 ballistic missile is a two-stage intercontinental ballistic missile (ICBM) believed to be capable of an effective range of 6000 to 6500 km, [12].

This puts Alaska and Singapore within range of this weapon. To intercept this weapon during boost phase, the intercepting missile must be fired from a platform within range of the ballistic missile launch site.

The TPD-2 has the following physical characteristics:

Parameter	**Magnitude**
Total Length = Stage 1 length + Stage 2 length + warhead/nosecone (m)	32
Range (km)	3500-4300
Payload (kg)	750-1000
Stage 1	
Length (m)	16
Diameter (m)	2.2
Stage 2	
Length (m)	14
Diameter (m)	1.335

Table 1. Characteristics(estimated) TPD-2 ICBM Data Input to Simulation(From [1])

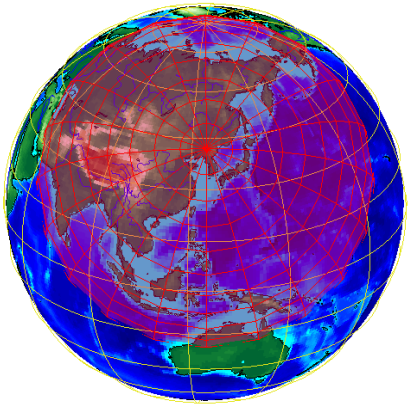

Figure 3. Reach of TPD-2 Missile

A 3DoF ballistic missile simulation model of the North Korean TPD-2 was used in this project. It consists of a series of MATLAB functions on an iterative integration loop, using 4 function files to model the TPD-2 missile:-

1. BRFlight3.m - integrates each time step to determine the current position,

attitude, and aerodynamic forces acting on the ballistic missile, generates a ballistic flight path of the ballistic missile target based on the model developed by Zarchan [6].;

2. BRParams3.m – determines ballistic missile mass and surface reference area;

3. BRDrag.m - determines target drag coefficient;

4. STatmos.m – determines the properties of the local atmosphere;

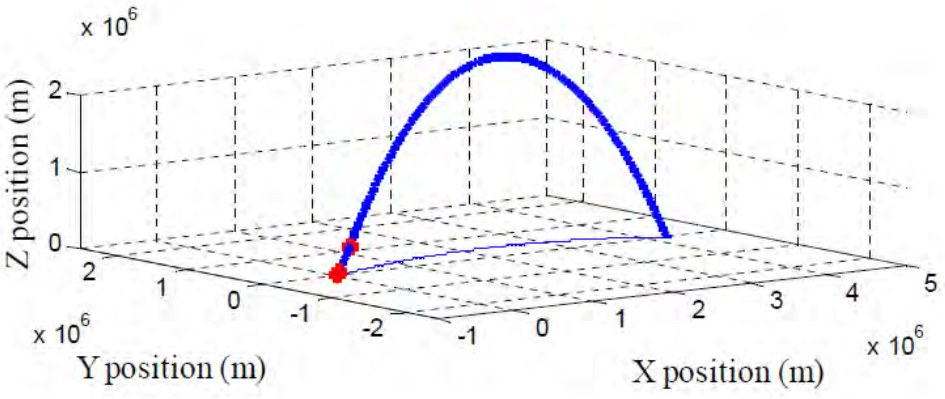

Figure 4. TPD-2 Ballistic Missile Flight Path simulated in MATLAB(From [1])

D. UCAV-LAUNCHED INTERCEPTOR MISSILE MODELING

The Northrop-Grumman X-47B [11] UCAV was used as the simulated launch platform for the interceptor missile. The X-47 project was initiated under the auspices of the J-UCAS program by DARPA. The X-47B is capable of high subsonic speeds (maximum speed estimated at Mach 0.9) and cruises at Mach 0.45, and has a service ceiling of greater than 40,000ft (12.2km). In addition, the X-47B has twin internal weapons bays conferring a 4,500lb-payload. Since a single AIM-120 air-to-air missile weighs 335 pounds [11], an X-47B will be able to carry at least 2 of them for combat persistence.

The X-47B has an endurance of greater than 6 hours and a maximum range of over 2000 miles (3200km). Because it remotely controlled, risk of injury to crew from enemy action or accidents is minimized and crew fatigue can be reduced as well. In addition, the platform is designed as a stealthy platform with a low RCS (exact figure

classified). Hence, it is ideal for autonomous and semi-autonomous Combat-Air-Patrol (CAP) missions near or even over hostile territory, making it closer to discovered or potential hostile ballistic missile launch sites.

The X-47B made its first flight on Feb 4, 2011, and flew on cruise configuration with its landing gear retracted on Sep 30, 2011. Testing is in progress to make the X-47B carrier-capable.

Figure 5. X-47B at take-off

Figure 6. X-47B at cruise

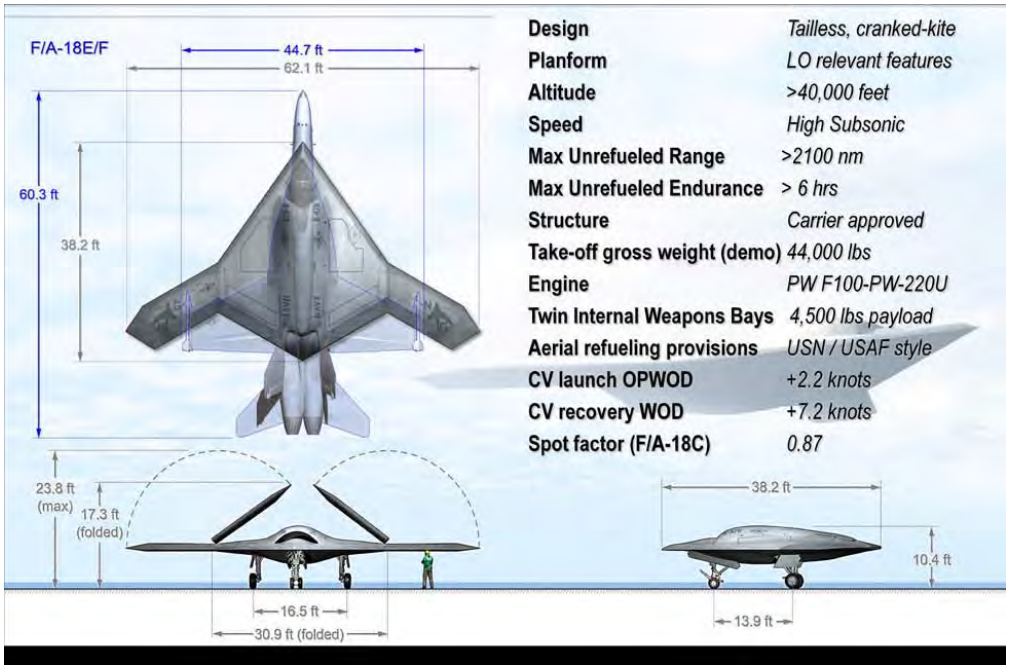

Figure 7. X-47B Specifications (From [11])

The interceptor missile, the Raytheon NCADE [9], an air-launched AIM-120-derivative missile designed to intercept threat ballistic missiles in the boost and ascent phases of flight. NCADE combines a Raytheon-proprietary (probably based on the AIM-9X focal plane staring array) seeker within an AIM-120 airframe and a booster stage. NCADE will be designed to intercept threat ballistic missiles in the endoatmosphere or exoatmosphere.

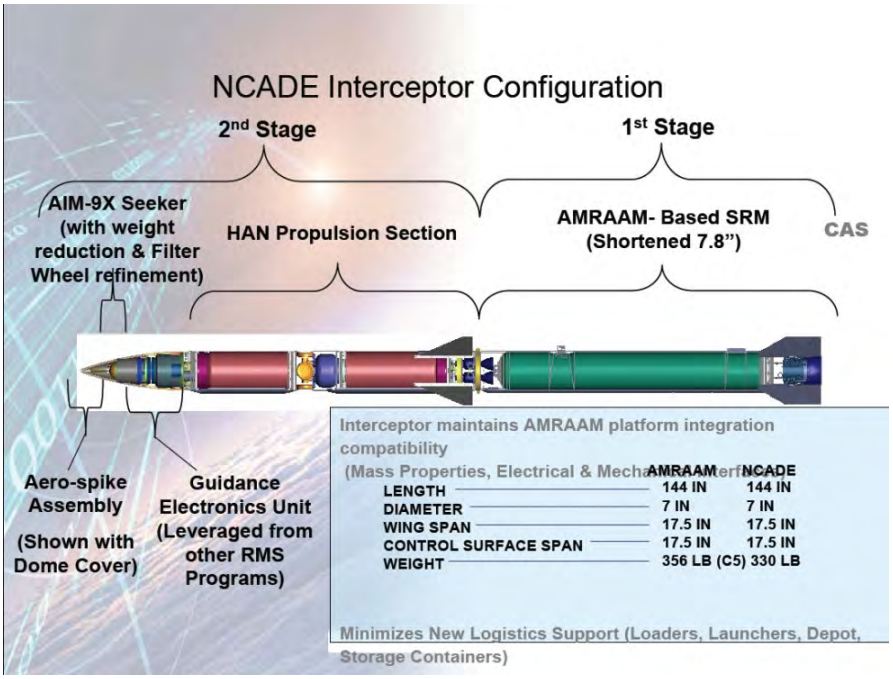

Figure 8. NCADE dimensions (From [9,10])

Figure 9. Interceptor Stage (Stage 2) (From [9])

Parameter	Magnitude
Total Length = Stage 1 length + Stage 2 length (including warhead/nosecone) (m)	3.66
Range (km)	250
Payload (kg)	18
Stage 1	
Length (m)	1.6529
Diameter (m)	0.1778
Stage 2	
Length (m)	2.0071
Diameter (m)	0.1778

Table 2. Characteristics(estimated) of NCADE Input into Simulation (From [8,9,10,11])

Because NCADE has the same form and fit as the AIM-120 [11], technically any aircraft that can carry the AIM-120 is also able to deploy the NCADE. A simulated model of the NCADE was utilized in this project based on publicly-available data from the US Navy and Raytheon, and if more information was required, assumptions were made on the specific capabilities. At no time was classified and/or proprietary data used in the simulations. The simulated interceptor missile was assumed to have a maximum range of 250km against a Ballistic Missile Target.

E. MISSILE FLIGHT PROGRAM

1. Initialization

As per previous research work done simulations [1,4,5], the Earth's geographical data required was based on the WGS84 values in the following references [14,15]:

Earth's Radius, R_e = 6,378,137 m

Earth's Semi-Minor Axis, b = 6,356,752 m

Earth's Flattening (1/Ellipticity), f = 1/298.257223563

Earth's Rotation Rate (Ωz_E) = 7.292116E-5 rad/sec

Earth's Gravitational Constant, GM = 3.986004418e14 m^3s^-2

2. Flight Program Description

From the NCADEFlightfromUCAV.m program initialization, it runs through several iterations. Upon integration of the previous iteration, it is returned to the program as the current values. The program then uses these values to calculate all the descriptive values of the system, apply the corrective time—and position—dependant factors, and calculate the derivatives for the next iteration.

The trajectory of the interceptor missile is generated by the function NCADETrajectory.m by applying the TS guidance law through the NCADEGuidance.m function. This is an iterative process, starting with a ‘guess’ of the interceptor final states and subsequently performing trajectory optimization to minimize the cost and penalty of each iterative flight path generated. The optimized flight path is then returned to NCADETrajectory.m for execution for a successful intercept.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TRAJECTORY SHAPING GUIDANCE AND ITS BENEFITS

In TS guidance, to derive the optimized interceptor flight path, a function J which consists of a set of two smaller functions, a Cost Function and Penalty functions, is to be minimized via multiple iterations [1,6]. The three parameters represented in the Cost Function are the length of the virtual arc (proportional to the flight path distance), the time to intercept and the impact angle of the final intercept (angular deviation from desired impact angle of 90deg). Each of the parameters needs to be weighted to correctly reflect the desired condition of the intercept, and affects the Cost Function value as a whole. (the default weighing for attaining the 90-deg impact angle is 1000).

$$J = t_{go^2} + W_{IA}(\mu - \mu_{desired})^2 + P_1 + P_2$$

The penalty function consists of the maximum acceleration in the y- and z-directions. They represent the 'penalty' to pay when certain physical limitations are attained or breached. The penalty function has been set to the certain values, which are a function of speed and altitude of the interceptor missile. They represent the physical limits beyond which the intercept solution will not be feasible.

The flight path optimization is done by solving a non-linear programming problem numerically real-time and once the minimum function is obtained, the algorithm will return the required control time history to the missile guidance system, which can then execute the commands and fly the derived flight path. Since the missile system can be programmed with sufficient data to compensate for its control system time constant, 25 the system lag can be effectively negated, thus eliminating a source of error. The guidance system can be updated every few seconds to increase the accuracy of the intercept.

The main advantages of this guidance are three-fold:(1) the relatively short computation time to iterate and converge to an optimized solution, (2) the cost and penalty functions are scalable as desired to fit the mission profile and (3) elimination of the control system time constant.

For a boost phase intercept mission, the TS guidance is able to address the disadvantages of the computed PN guidance and offers greater flexibility in being able to ‘customize’ the guidance to improve its performance to meet the different operational demands.

IV. SIMULATED LAUNCHES AND RESULTS

The NCADE-derived interceptor missile was simulated as being air-launched by an X-47B UCAV at 10km (32808ft) of altitude, at an initial airspeed of 100m/s (approximately Mach 0.33 for a speed of sound of 299.5m/s) and zero-pitch angle. This a possible typical combat-air-patrol (CAP) flight condition by an X-47B UCAV.

The purpose of this simulation was to determine if the Direct-Methods TS guidance law [1], already proven in simulation for vertical surface launched interceptors [4,5], is equally applicable to interceptor missiles launched from level flight (zero-pitch angle) at altitude.

A. SIMULATION RESULTS (LAUNCH ALT = 10KM, LAUNCH AIRSPEED=100M/S OR MACH 0.33)

The Northing and Westing distances were both 100km (default) and the weighing (from 90deg impact angle) assigned was 1000(default). The interceptor missile is gaining altitude and aligns itself to hit the target ICBM at an angle close to 90deg for maximum kinetic energy. The Euler angles of the interceptor missile are shown in Figure 11. Figure 12 illustrates the velocity profile from launch to target interception. The interceptor missile accelerates steadily, showing a almost-proportional increase in velocity to a peak velocity at about t=20s, then the velocity stabilizes at about t=35s before finally hitting the target at t=42s. At no point was there any unsteady velocity fluctuations during the endgame, thus showing the ability of the interceptor guidance mechanism to achieve an optimal trajectory.

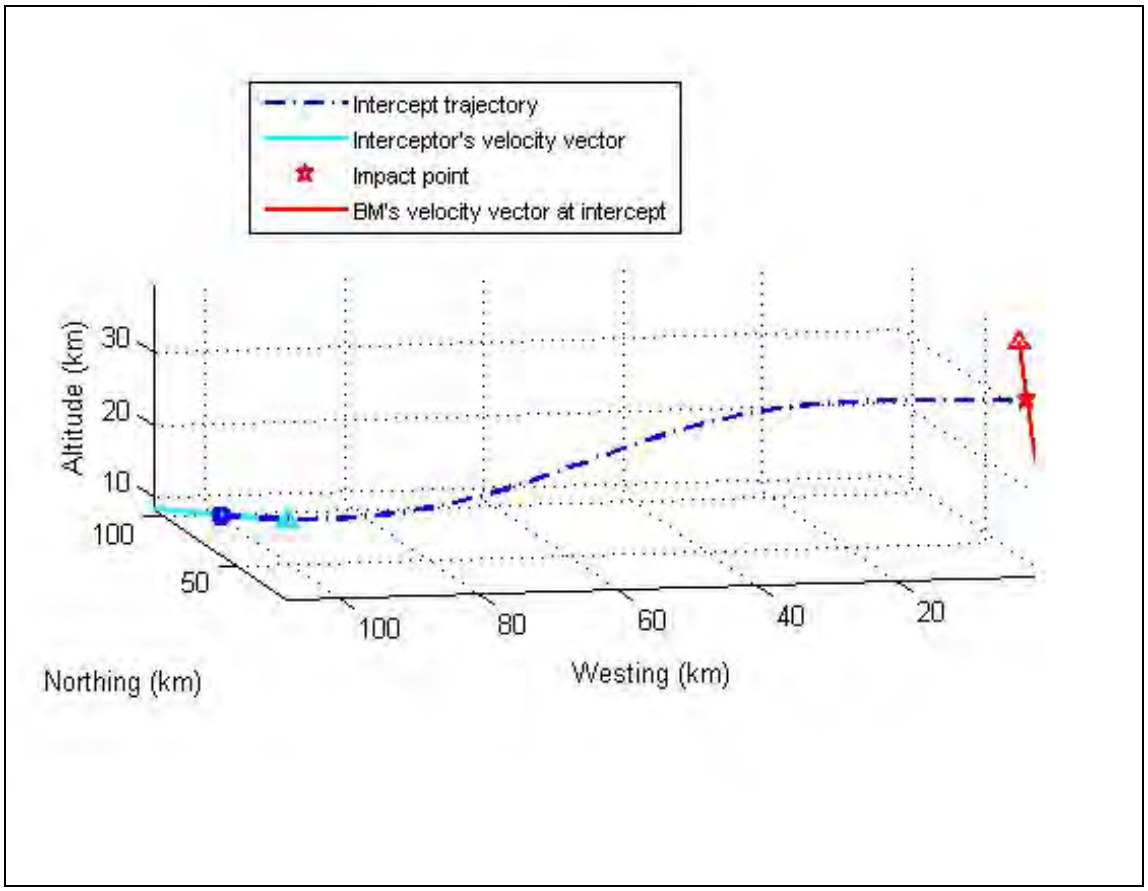

Figure 10. Interceptor Trajectory at Default Conditions (Northing: 100km, Westing: 100km, Weighing:1000)

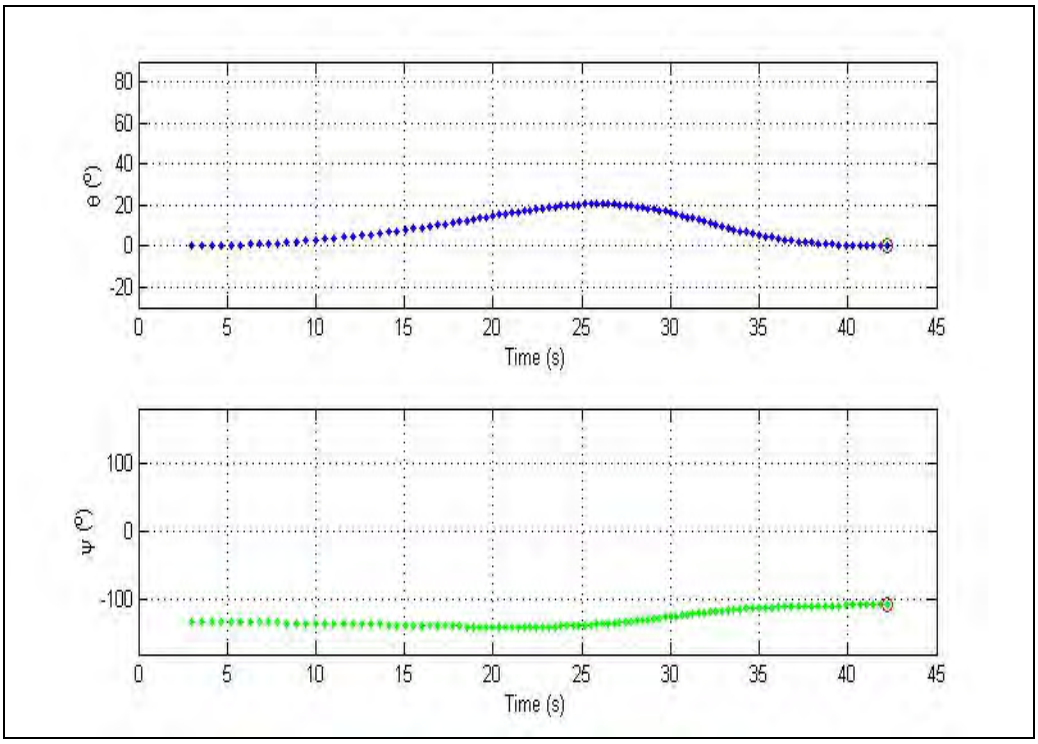

Figure 11. Interceptor Euler Angles

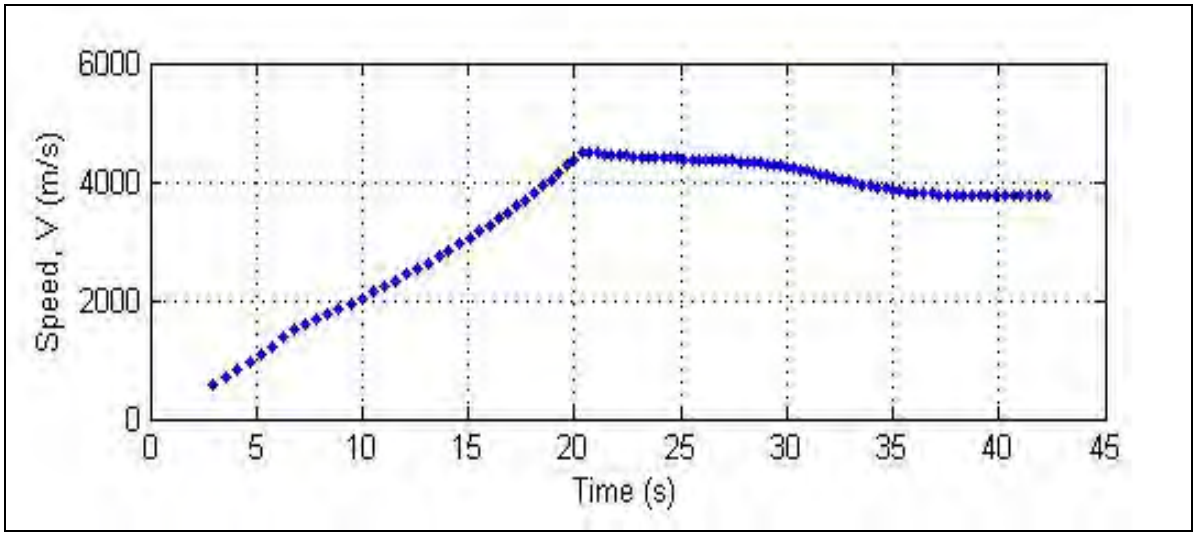

Figure 12. Interceptor Trajectory at Default Conditions (Northing: 100km, Westing: 100km, Weighing:1000)

After optimization, the G-load factor constraints are satisfied (ie. no or negligible dynamic constraint violations in lateral accelerations) Figure 13 illustrates this. In addition, Figure 14 shows how the impact angle was adjusted during the optimization. In addition, it can be observed that the estimate of tgo converges to its steady-state value after about 60 iterations.

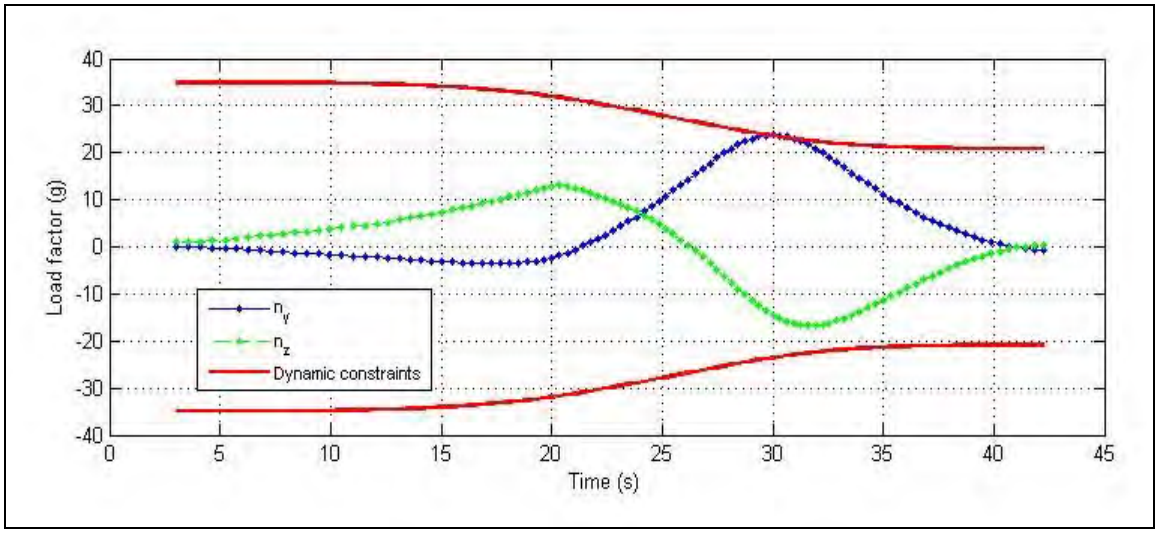

Figure 13. Lateral Acceleration (LATAX, Ny and Nz) Load Factor (No or negligible dynamic constraint violations

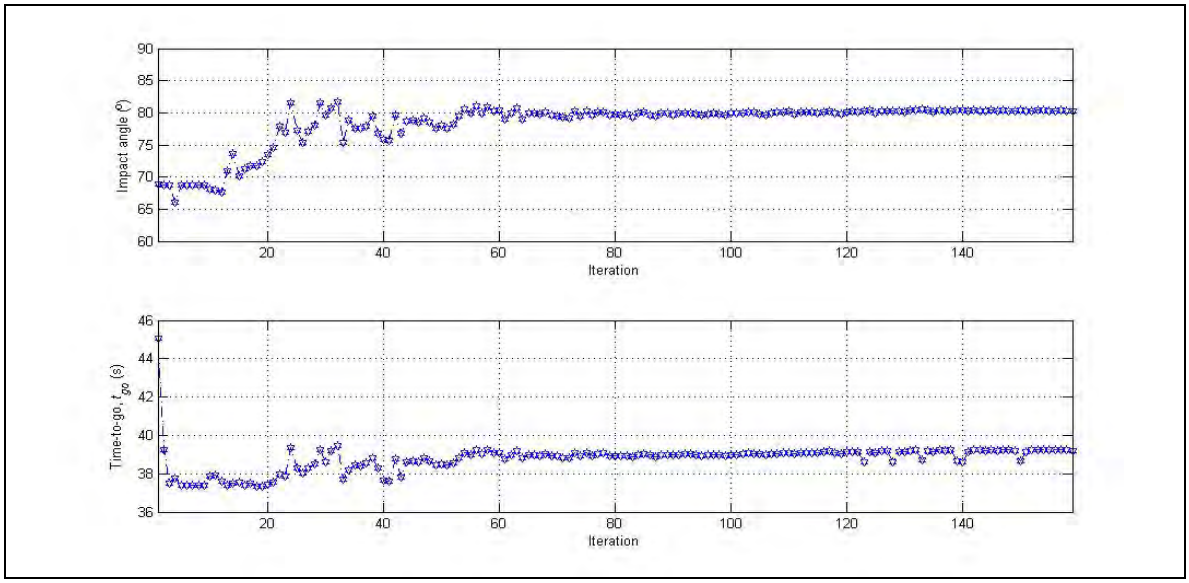

Figure 14. Interceptor Missile Impact Angle and Time-to-go

B. LAUNCH ALT = 17KM, LAUNCH AIRSPEED=100M/S

After the initial default simulations looked promising, it was decided to increase the launch altitude to 17km to determine the kinematic boundary of the interceptor missile at a higher altitude. The Kinematic Boundary (KB) of the interceptor missile can be regarded as the locus of the points representing the maximum range representing the maximum range at which the ballistic missile target may be successfully engaged as a function of relative bearing from the target at the start of the engagement.

Several simulated interceptor launches were made to generate a composite trajectory plot as a means of determining the KB, as shown in Figure 15. The individual interceptor trajectories are 3-dimensional and thus not planar.

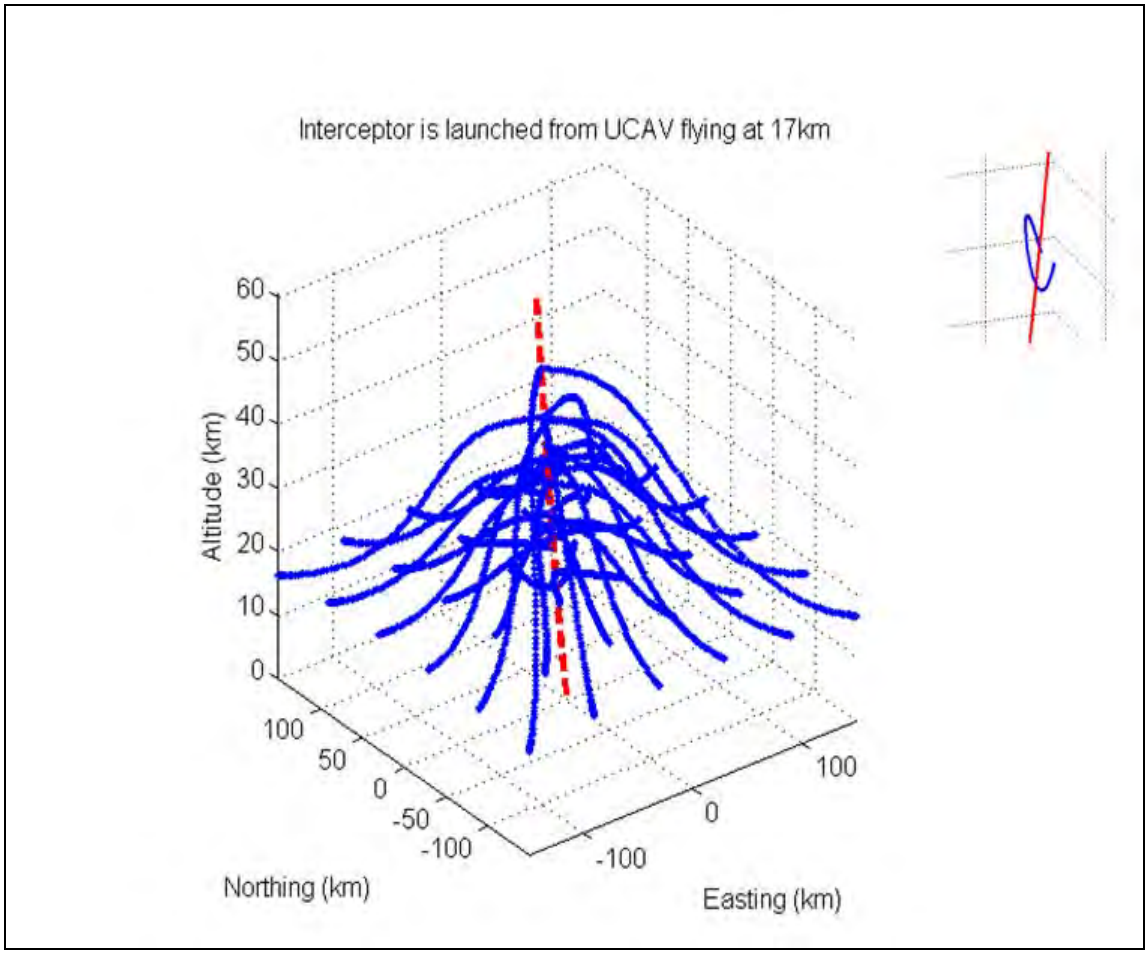

Figure 15. Composite Interceptor Trajectory from Different Directions (the individual interceptor trajectories are 3D)

Figure 16 demonstrates that as long as the launch is carried out within a 60km radius from the interceptor missile launcher, the interceptor missile will hit the target. Hence, at a launch altitude of 17km, the interceptor missile's kinematic boundary is 60km. For optimal "hit-to-kill" kinetic energy, an impact angle as to or exactly at 90 deg is required. The plots show that an impact angle of 80 deg or more occur within 60km.

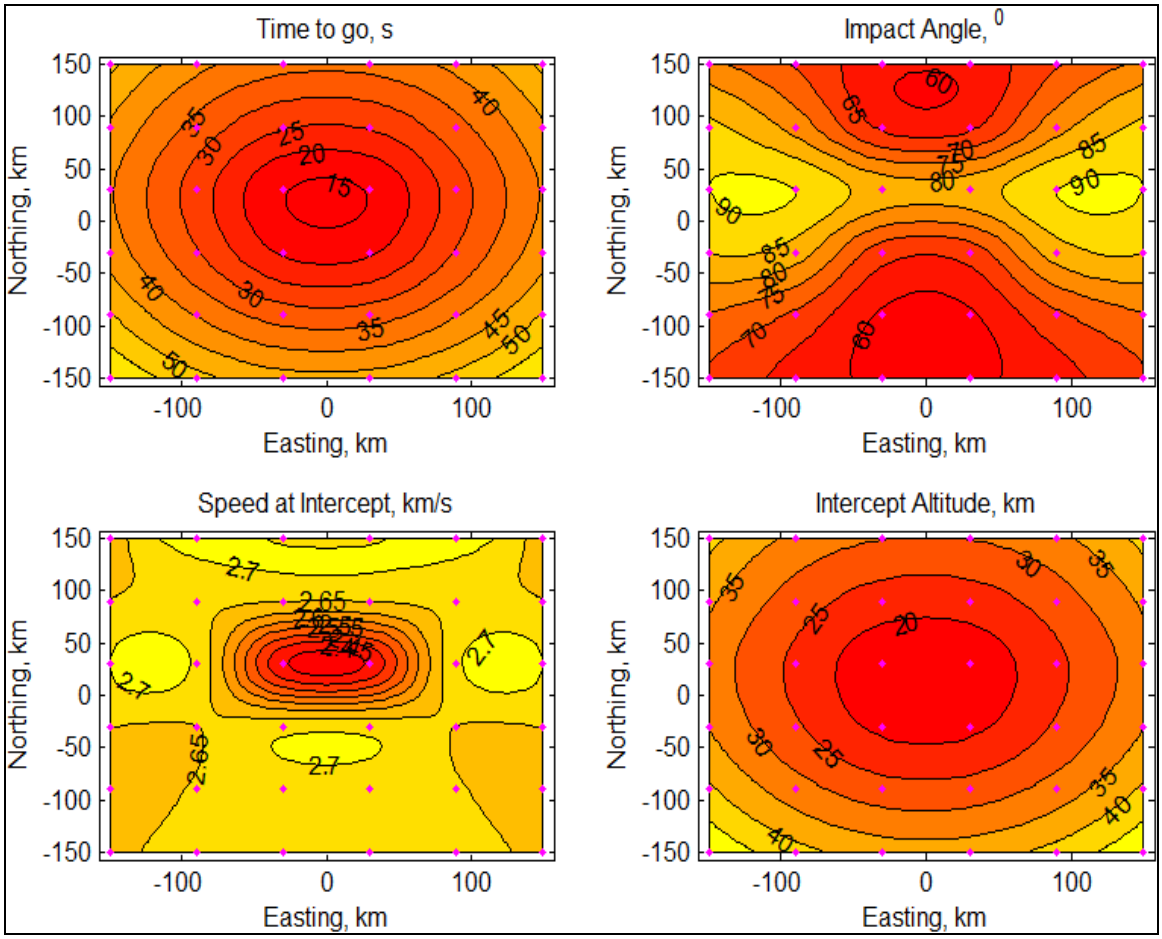

Figure 16. Kinematic Boundary (max radius 60km) of Interceptor Missile at launch alt =17km (all constraints satisfied)

In addition, Figure 17 illustrates that the lateral acceleration constraint violation in the Z-direction (Nz) is zero G and in the Y-direction (Ny) of the interceptor missile is negligible at less than 0.5G. The scaled combined performance index plot in Figure 17 mirrors the impact angle plot of Figure 16, illustrating the correlation between the performance index and the high weighing of the 90 degree impact angle requirement on the optimization.

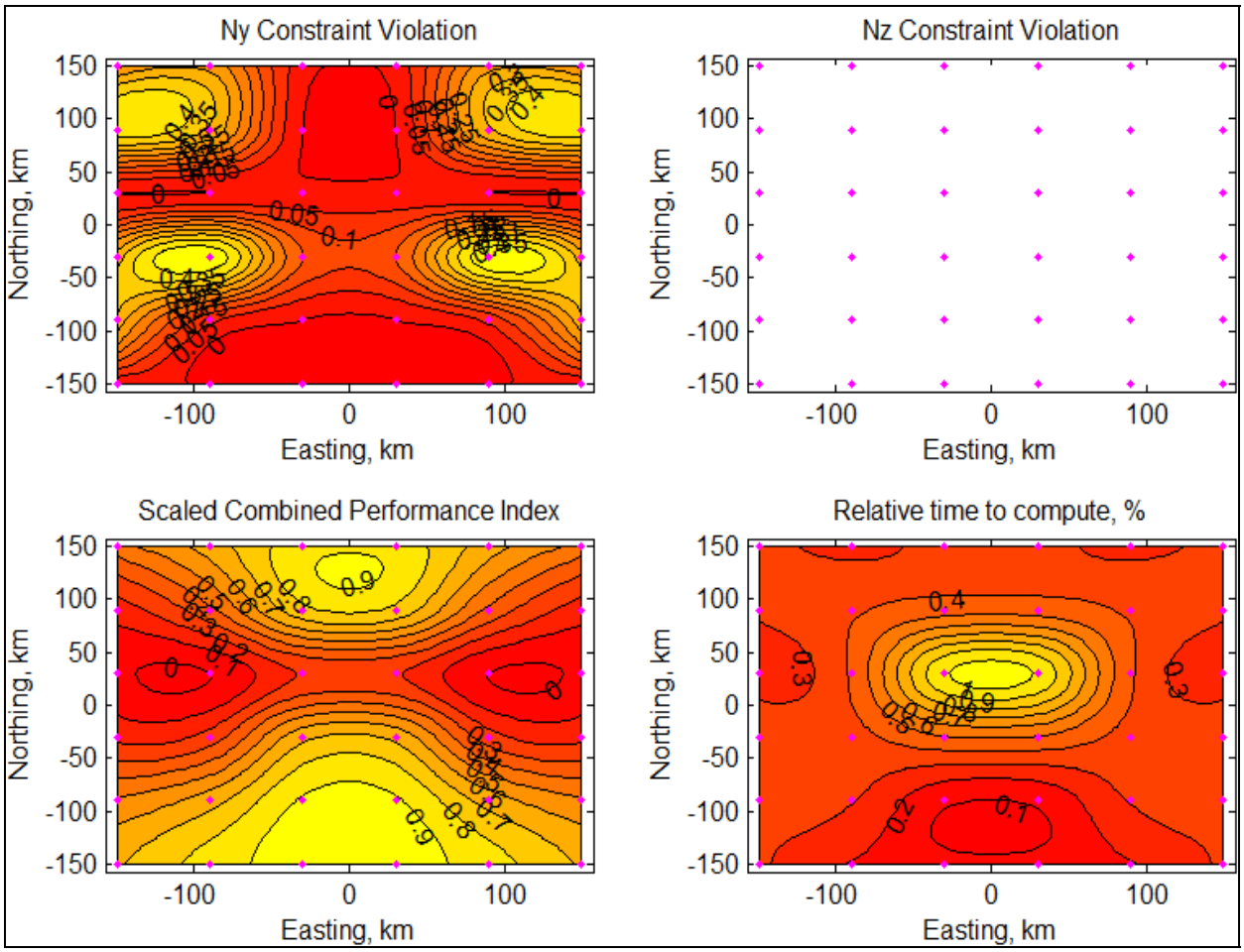

Figure 17. Kinematic Boundary(max radius 60km) of Interceptor Missile at launch alt =17km (additional plots)

THIS PAGE INTENTIONALLY LEFT BLANK

V. DISCUSSIONS, ADDITIONAL RESULTS AND ANALYSIS

After the initial simulations described in the previous chapter were conducted, it was decided to vary the interceptor launch altitude to evaluate the flexibility of the algorithm at different launch altitudes. Figure 18 illustrates the trajectory comparison with launch at a lower altitude compared to higher altitude.

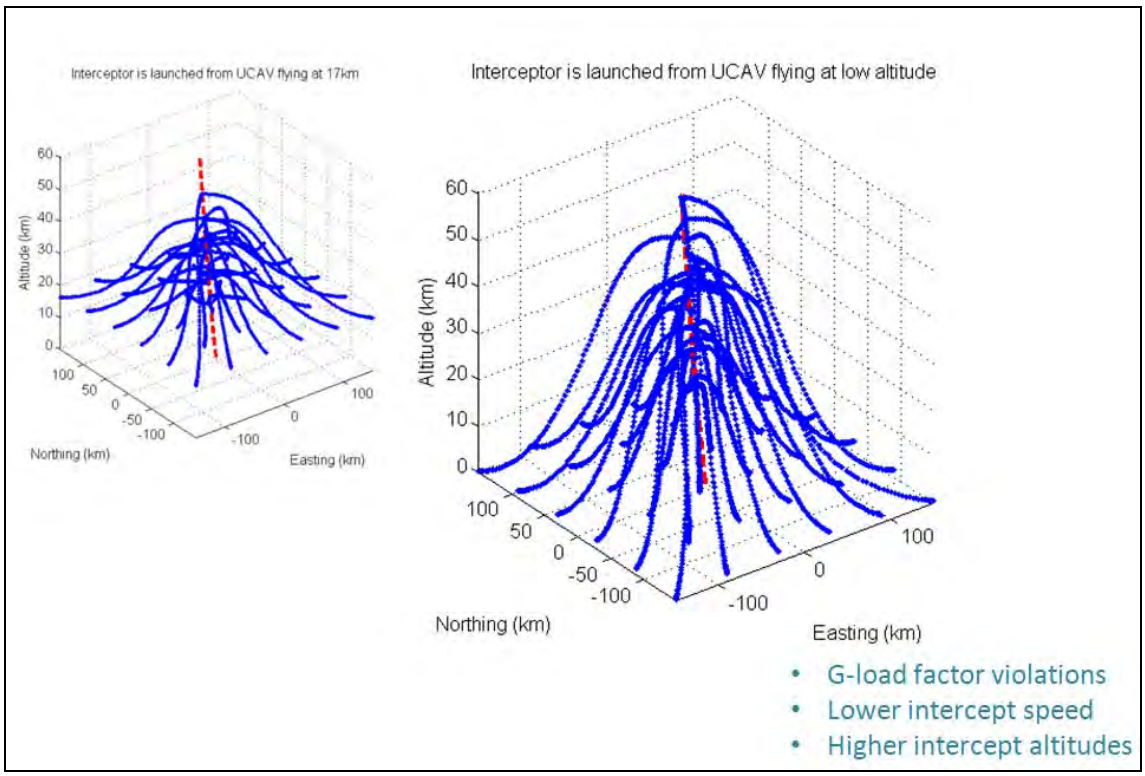

Figure 18. Trajectories at a lower launch altitude of 10km compared to the original launch altitude of 17km

At lower altitudes , while keeping the weighing for impact angle at 90deg at a high of 1000, it was observed that lateral acceleration (LATAX) G-load factor (Ny and Nz) violations increased and interception of the Ballistic Missile Target took place at lower speeds and higher intercept altitudes (ie. Forcing the missile to fly high). Removing the 90deg impact angle requirement by making the weighing 0 allows the interceptor missile to be air-launched from any altitude (see Figure 19).

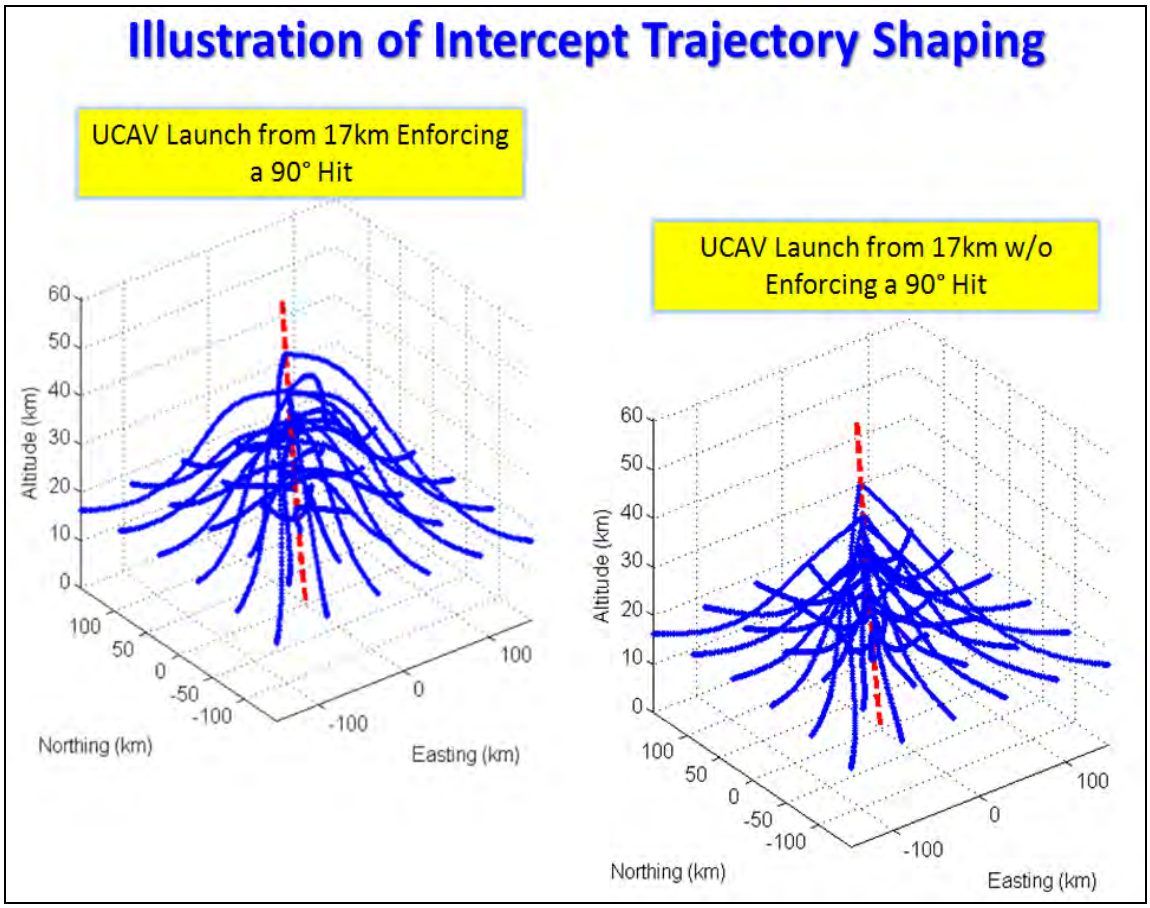

Figure 19. Interceptor Trajectories at launch altitude of 17km (90deg Impact Angle weighing and non-weighing comparison)

Figure 20 shows that even with weighting requirement for 90 deg impact angle removed, the kinematic boundary is increases to ~150km. In addition, the simulated impact angles are not much lower than the previous case (see Figure 16 for comparison).

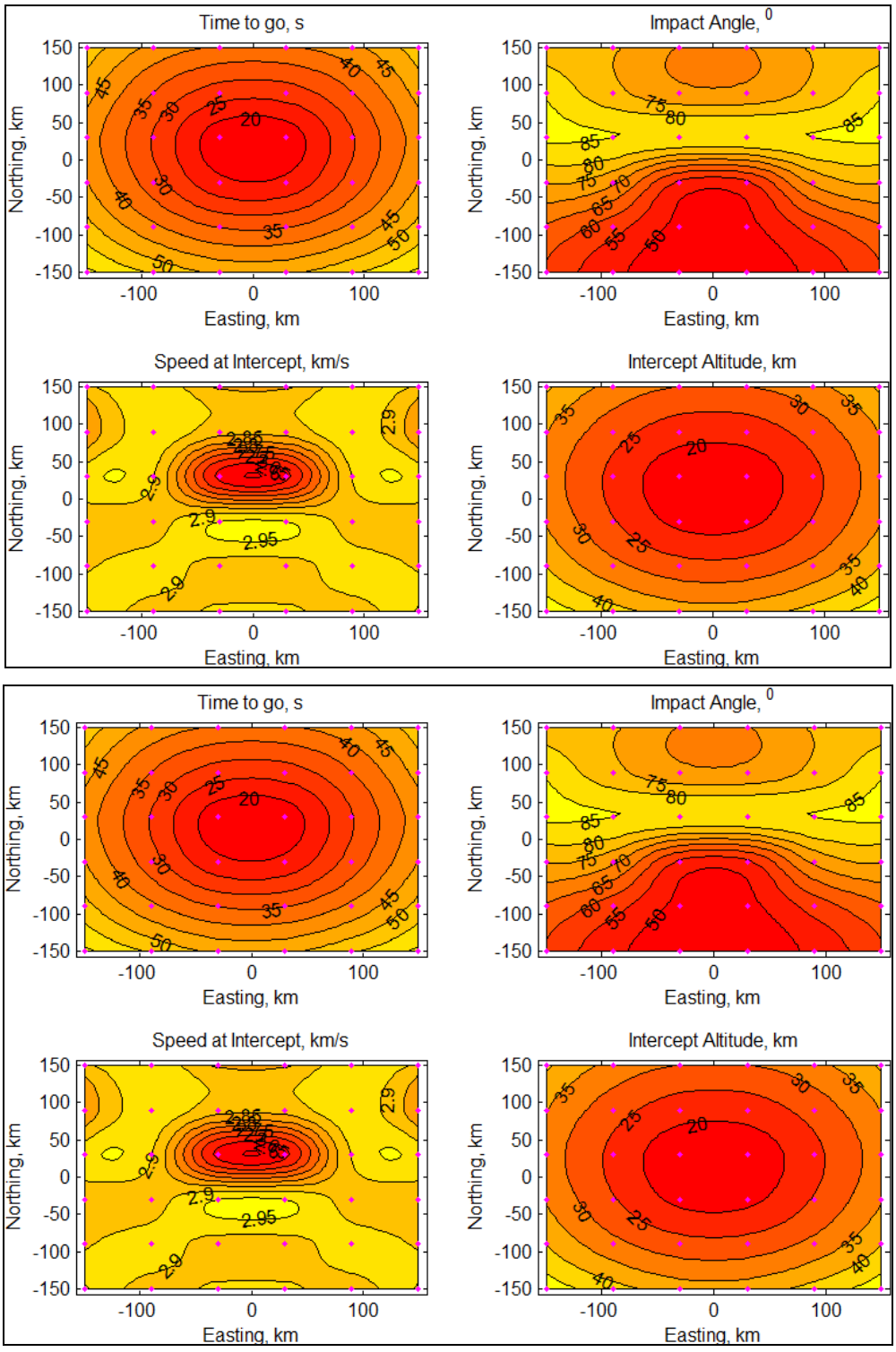

Figure 20. Boundary increased to 150km with new launch altitude of 10km and 90deg Impact Angle requirement

In Figure 21, the scaled-performance index is better (ie. of a lower magnitude) and a 0-index covers a much larger area compared to the previous case, reinforcing the larger kinematic boundary conferred by removing the 90-degree impact angle requirement. In addition, lateral acceleration dynamic constraint violations are zero.

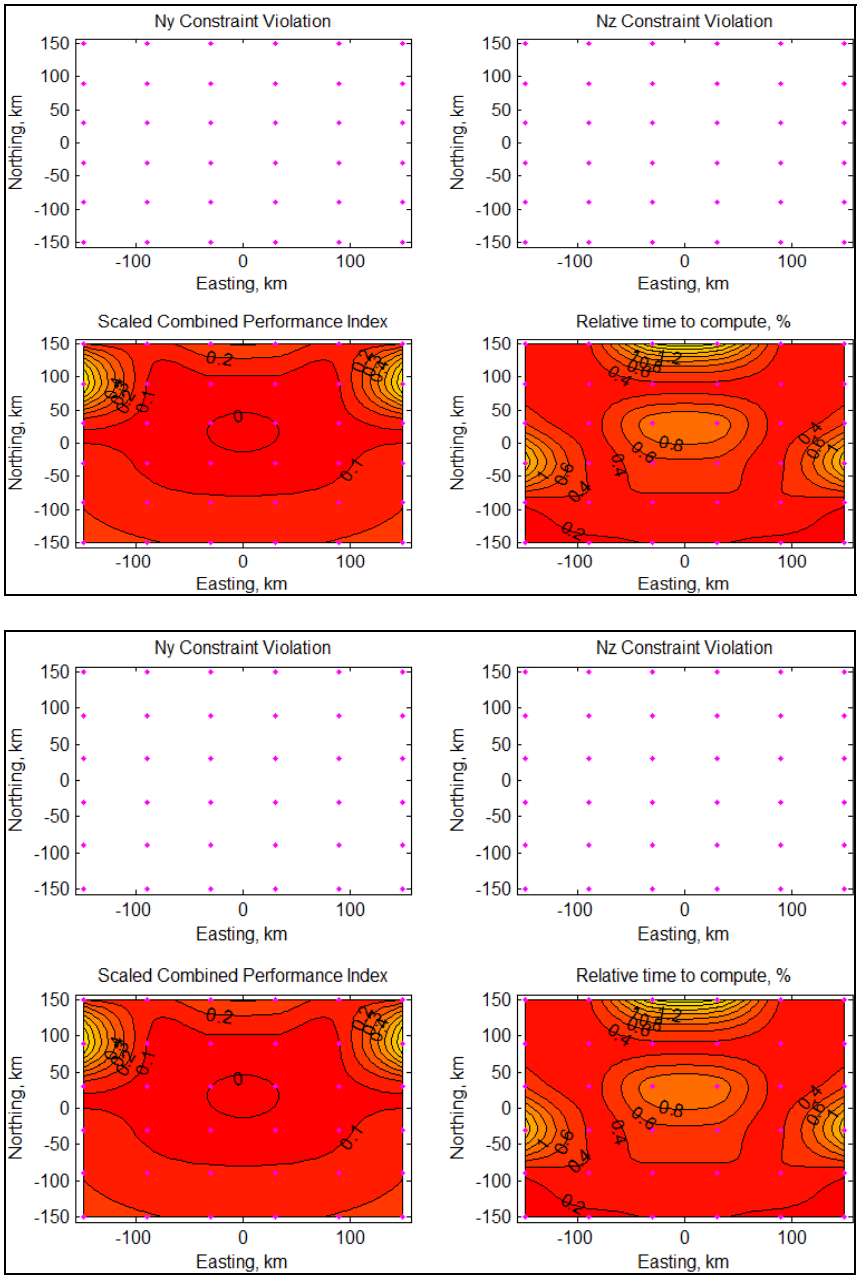

Figure 21. Kinematic Boundary increased to 150km with new launch altitude of 10km and 90deg Impact Angle requirement removed (additional plots)

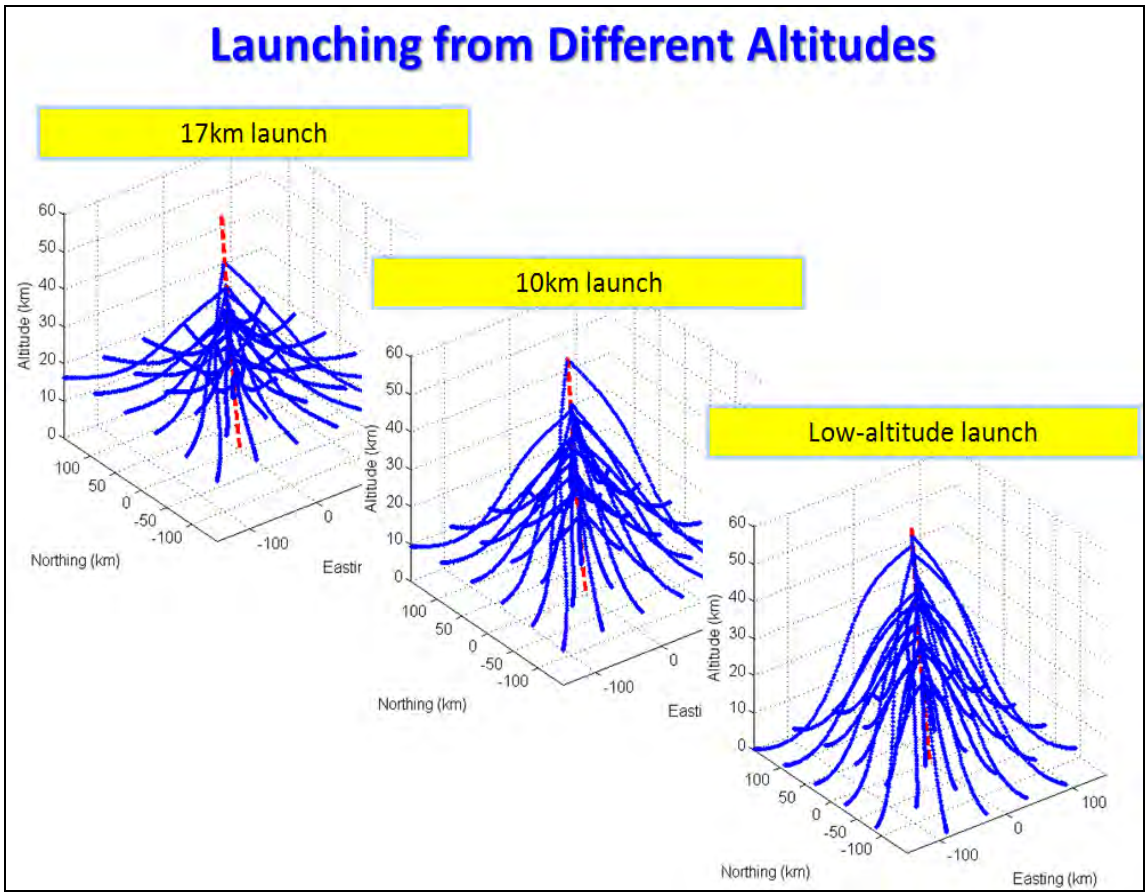

Figure 22. Various launch altitudes compared (weighing for 90 deg impact angle removed)

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

This project has shown that it is feasible to use Directed-Method Trajectory Shaping in air-launched anti-ballistic missiles from UCAVs in addition to launches from land and surface platforms. For greater accuracy, future work would be to investigate the effects of increasing the time-delay from ballistic missile target launch to interceptor missile carrier platform detection and launch of interceptor missile. In addition, a higher-fidelity UCAV model could be developed and integrated with the model.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Lukacs IV, John and Yakimenko, Oleg, "Trajectory-Shaping Guidance for Interception of Ballistic Missiles in the Boost Phase," *Journal of Guidance,Control, and Dynamics*, Vol 31, No. 5, September–October 2008, pp 1524–1531.

[2] Yakimenko, O.A., "Direct Methods for Rapid Prototyping of Near-Optimal Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol 23, No. 5, September-October 2000, pp 865-875

[3] Yakimenko, O.A., AE4902 "Direct Method of Calculus of Variations as the Means for Rapid Prototyping of Optimal Trajectories" Course Notes, Spring 2011.

[4] Lukacs IV, J., "New Missile Guidance Algorithm For The Interception Of Ballistic Missiles In The Boost Phase," Master's thesis, Naval Postgraduate School, Monterey, CA, 2006.

[5] Leong, W.W. "Trade-off Study for the Hit-to-kill Interception of Ballistic Missiles in the Boost Phase" Master's thesis, Naval Postgraduate School, Monterey, CA, 2009.

[6] Zarchan, P., "Tactical and Strategic Missile Guidance Fourth Edition," Vol. 199, American Institute of Aeronautics and Astronautics, Reston, VA, 2002

[7] Siouris, G. M., *Missile Guidance and Control Systems, 1st Edition*, Springer-Verlag, New York, Inc, 2004.

[8] http://www.raytheonmissiledefense.com/phases/index.html Raytheon, "Missile Trajectory Phases"

[9] www.raytheon.com/newsroom/feature/ncade_07-09/ (NCADE, an air-launched AIM-120 AMRAAM derivative fitted with a modified AIM-9X imaging IR seeker for use against ballistic missiles)

[10] www.raytheon.com/capabilities/products/stellent/groups/rms/documents/content/cms01_054563.pdf (AIM-120 AMRAAM brochure with missile physical dimensions)

[11] www.as.northropgrumman.com/products/nucasx47b/assets/UCASD_DataSheet_final.pdf (X-47B UCAV data)

[12] www.fas.org/programs/ssp/nukes/nuclearweapons/Taepodong.html (Federation of American Scientists (FAS) website description of a type of ballistic missile believed to be in North Korean service)

[13] www.af.mil/information/factsheets/factsheet_print.asp?fsID=102&page=1 (Boeing F-15 fighter data. The F-15 is the planned launch platform for the test launch of AIM-120-derived NCADE missiles for use against ballistic missile targets)

[14] Hutchins, R., ME4703 "Missile Flight Analysis" Course Notes, Spring 2005.

[15] Stevens, B., and Lewis, F., *Aircraft Control and Simulation Second Edition*, John Wiley and Sons, 2003.

APPENDIX

A. NCADEFLIGHTFROMUCAV.M

Description:

This MATLAB code develops and tracks the trajectory of the interceptor missile. It was originally written for a surface-launched interceptor, but subsequently modified by the author to replicate an air-launched interceptor.

```
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006
% Corrected by O.Yakimenko, August 2007, November 2009
% Modified by Zheng Liang Lu, Singapore on May 2011

% This script develops and tracks the flight path of the interceptor
% missile. For the first ten seconds it integrates a series of
% acceleration commands to simulate an airborne, near-vertical launch
and tip-over.
% Upon activation of the guidance law, it sends the known values to the
guidance
% law and receives back the future time history of the optimal flight
path.
% This script then implements that optimal path. It updates the final
% conditions and recalculates the optimal flight path at an interval of
10
% seconds. The script calls BRFlight3, NCADEParams1.m, NCADEDrag.m,
STatmos.m,
% and NCADEGuidance.m

%% List of variables
% Acc_SM     = index-based vector of acceleration values
% AllSM      = index based vector of all interceptor values
% alt        = altitude
% CD         = drag coefficient
% count      = counting variable to determine guidance law update
interval
% dist       = cumulative distance travelled
% Drag       = total drag force
% Forces_SM  = index-based vector of force values
% g          = gravitational force, based on WGS-84 value of
gravitational
%              attraction and altitude
% m_i        = interceptor mass
% Model_SM   = index-based vector of internal values
% MV         = interceptor speed in Mach (relative to local speed of
sound)
% N1,N2      = variables used to ensure optimal vectors are the same
length
% num_SM     = number of interations conducted (used for plotting)
```

```
% nx,ny,nz   = axial acceleration command, body frame x, y and z,
respectively
% nx(y,z)_Op = index based vector of the optimal flight path values
% path       = returned time history of the optimal path
% Pos_SM     = index-based vector of position values
% press      = local atmospheric pressure
% psi        = heading angle
% psidot     = rate of change of heading angle
% psidot_Op  = index based vector of the optimal flight path values
% px,py,pz   = x, y and z components of position
% px(y,z)_Op = index based vector of the optimal flight path values
% py_Op      = index based vector of the optimal flight path values
% pz_Op      = index based vector of the optimal flight path values
% q          = counting variable (used in another program)
% Re         = WGS-84 Earth's radius
% ro         = local atmospheric density
% Sref       = planar reference area (for drag calculations)
% state      = the state of the interceptor
% t          = global time
% target     = the state of the rocket
% temp       = local atmospheric temperature
% tgo        = time to go to intercept
% th         = flight path angle
% th_Op      = index based vector of the optimal flight path values
% thdot      = rate of change of flight path angle
% thdot_Op   = index based vector of the optimal flight path values
% Thrust     = thrust generated by interceptor motor
% time_Op    = index based vector of the optimal flight path values
% time_SM    = index-based vector of time values
% update     = number of updates to the guidance law conducted
% V          = velocity of the interceptor
% V_Op       = index based vector of the optimal flight path values
% Vdot       = rate of change of the velocity
% Vdot_Op    = index based vector of the optimal flight path values
% Vel_SM     = index-based vector of velocity values
% w          = number of 0.5s steps between the lunches of traget and
interceptor

close all, clear all, clc

global Re alphag path            % shared by SMFlight3, NCADEGuidance &
NCADETrajectory
global q states                  % shared by SMFlight3 & NCADEGuidance
global Pos_BR Pos_SM Npol Npt kpolar weight90 % ... by SMFlight3 &
NCADETrajectory

%% Computing the flight path of a Ballistic Missile in a gravity turn
BRFlight3

%% Initializing variables for an Interceptor
t=0; dt=0.5; q=0; w=120; % 60 sec detection time
Nupd=30; count=Nupd; update=0;
Npt=100;            % the number of points the optimal trajectory is
computed at
```

```
Npol=8;             % number of coefficients in approximating
polynomials
kpolar=0.02;

 prompt = {'Northing wrt to BM launch point, km',...
           'Westing wrt to BM launch point, km',...
           'Weighting coefficient for impact angle'};
 dlg_title = 'Enter Interceptor launch coordinates';
 num_lines = 1;
 def = {'100','100','1000'};
 answer = inputdlg(prompt,dlg_title,num_lines,def,'on');
 px      =str2num(answer{1})*1000;
 py      =str2num(answer{2})*1000;
 weight90=str2num(answer{3});
% px=-100000/20;
% py=100000/2;
% weight90=100;
% pz=sqrt((Re)^2-px^2-py^2);
%assume an initial launch alt of 10000m (updated May 2011 for UCAV
launch, further amended Oct 2011)
pz=sqrt((Re+10000)^2-px^2-py^2);
px_old=px; py_old=py; pz_old=pz;
dist=0;

psi=atan2(Pos_BR(120,2)-py,Pos_BR(120,1)-px); psi_old=psi;
%th=90*pi/180; th_old=th;
%%changing the original pitch angle 'th' from 90deg to another
value>=30deg
%%pitch angle 'th' seems to be of a minimum of ~10deg for an initial
10000m alt
th=0*pi/180; th_old=th;
%th=0*pi/180; th_old=th;
%V=1; V_old=V;
%%changing the original initial velocity V from 1 m/s to an assumed
value
%%based on the UCAV flight profile at an inital alt
V=100; V_old=V;

%% Computing Interceptor flight path
for i=1:20%1000
    t=t+dt;
%% Boost phase (angled launch, was originally vertical launch), t<10s
if t<3
    % Speed, Mach number

    alt=norm([px;py;pz])-Re;
    [ro,press,temp]=STatmos(alt);
    MV = V/sqrt(1.402*287.053*temp);

    % Forces
    g=3.986004418e14/norm([px;py;pz])^2;
    [m_i,Sref] = NCADEParams1(t);
    CDtable = NCADEDrag(MV);
        if t<=6,    Thrust = 206000; psidot=0;        thdot=0;
```

```
        else        Thrust = 95300;  psidot=0;        thdot=0;
end
        %%original line below:-
        %else        Thrust = 95300;  psidot=0;        thdot=-0.075;
end
    ny = V/g*cos(th)*psidot;
    nz = V/g*thdot+cos(th);
    nn = sqrt(ny^2+nz^2);
        CL=(2*nn*m_i*g)/(ro*V^2*Sref);
        CD = CDtable(1) + kpolar*CL^2;
    Drag = ro*V^2*CD*Sref/2;
    nx=(Thrust-Drag)/m_i/g;

    alphag=180/pi*nn*m_i*g/(ro*V^2*Sref/2)/13;

    % Kinematics
    Vdot=g*(nx-sin(th));
    psidot=ny*g/V/cos(th);
    thdot=g*(nz-cos(th))/V;

    % Collecting variables
    time_SM(i,1)=t;
    Forces_SM(i,1)=nx;     Forces_SM(i,2)=ny;     Forces_SM(i,3)=nz;
    Model_SM(i,1)=V;       Model_SM(i,2)=Vdot;
    Model_SM(i,3)=th;      Model_SM(i,4)=thdot;
    Model_SM(i,5)=psi;     Model_SM(i,6)=psidot;
    Pos_SM(i,1)=px;        Pos_SM(i,2)=py;        Pos_SM(i,3)=pz;
    Pos_SM(i,4)=dist;
    Vel_SM(i,1)=V*cos(th)*cos(psi);
    Vel_SM(i,2)=V*cos(th)*sin(psi);
    Vel_SM(i,3)=V*sin(th);
    Acc_SM(i,1)=Vdot*cos(th)*cos(psi)-V*cos(th)*sin(psi)*psidot...
        -V*sin(th)*cos(psi)*thdot;
    Acc_SM(i,2)=Vdot*cos(th)*sin(psi)+V*cos(th)*cos(psi)*psidot...
        +V*sin(th)*sin(psi)*thdot;
    Acc_SM(i,3)=Vdot*sin(th)+V*cos(th)*thdot;

    % Euler integration
    V=V_old+Vdot*dt;
    psi=psi_old+psidot*dt;
    th=th_old+thdot*dt;
    px=px_old+V*cos(th)*cos(psi)*dt;
    py=py_old+V*cos(th)*sin(psi)*dt;
    pz=pz_old+V*sin(th)*dt;

    dist=(dist+abs(norm([px-px_old;py-py_old;pz-pz_old])));

    V_old=V;      psi_old=psi;  th_old=th;
    px_old=px;    py_old=py;    pz_old=pz;

else
%% Optimal guidance, t>10s
    if count==Nupd & update==0     % Recomputing the trajectory every
Nupd cycles
```

```
        update=update+1;
        fprintf('Starting Interceptor''s Guidance Update
#%2.0f\n',update)
        state=[px;py;pz;V;th;psi;Vdot;thdot;psidot];
        target=[Pos_BR(i+w,1);Pos_BR(i+w,2);Pos_BR(i+w,3);
                Vel_BR(i+w,1);Vel_BR(i+w,2);Vel_BR(i+w,3);
                Acc_BR(i+w,1);Acc_BR(i+w,2);Acc_BR(i+w,3)];
        path=NCADEGuidance(t,state,target,i);      % Calling
NCADEGuidance function
            if update==1;
                N1=length(path(:,1)); N2=N1;
            else
                N2=length(path(:,1));
            end
        % Identifying variables
        time_Op(:,update)=[path(:,1);zeros(N1-N2,1)];
        px_Op(:,update)=[path(:,2);zeros(N1-N2,1)];
        py_Op(:,update)=[path(:,3);zeros(N1-N2,1)];
        pz_Op(:,update)=[path(:,4);zeros(N1-N2,1)];
        V_Op(:,update)=[path(:,5);zeros(N1-N2,1)];
        th_Op(:,update)=[path(:,6);zeros(N1-N2,1)];
        psi_Op(:,update)=[path(:,7);zeros(N1-N2,1)];
%         Vdot_Op(:,update)=[path(:,8);zeros(N1-N2,1)];
%         thdot_Op(:,update)=[path(:,9);zeros(N1-N2,1)];
%         psidot_Op(:,update)=[path(:,10);zeros(N1-N2,1)];
        nx_Op(:,update)=[path(:,8);zeros(N1-N2,1)];
        ny_Op(:,update)=[path(:,9);zeros(N1-N2,1)];
        nz_Op(:,update)=[path(:,10);zeros(N1-N2,1)];
        count=1;
    end
            if     count==101            % added by OY 2009-10-08
                   count=count-1;
            elseif count==0
                   count=1;
            end

    t=time_Op(count,update);
    nx=nx_Op(count,update);
    ny=ny_Op(count,update);
    nz=nz_Op(count,update);
    V=V_Op(count,update);
%     Vdot=Vdot_Op(count,update);
    th=th_Op(count,update);
%     thdot=thdot_Op(count,update);
    psi=psi_Op(count,update);
%     psidot=psidot_Op(count,update);
    px=px_Op(count,update);
    py=py_Op(count,update);
    pz=pz_Op(count,update);

    % Collecting variables
    time_SM(i,1)=t;
    Forces_SM(i,1)=nx;     Forces_SM(i,2)=ny;     Forces_SM(i,3)=nz;
    Model_SM(i,1)=V;       Model_SM(i,2)=Vdot;
    Model_SM(i,3)=th;      Model_SM(i,4)=thdot;
```

```
    Model_SM(i,5)=psi;     Model_SM(i,6)=psidot;
    Pos_SM(i,1)=px;        Pos_SM(i,2)=py;        Pos_SM(i,3)=pz;
    Pos_SM(i,4)=dist;
    Vel_SM(i,1)=V*cos(th)*cos(psi);
    Vel_SM(i,2)=V*cos(th)*sin(psi);
    Vel_SM(i,3)=V*sin(th);
    Acc_SM(i,1)=Vdot*cos(th)*cos(psi)-V*cos(th)*sin(psi)*psidot...
        -V*sin(th)*cos(psi)*thdot;
    Acc_SM(i,2)=Vdot*cos(th)*sin(psi)+V*cos(th)*cos(psi)*psidot...
        +V*sin(th)*sin(psi)*thdot;
    Acc_SM(i,3)=Vdot*sin(th)+V*cos(th)*thdot;
    Update(i,1)=update;

    % Time step
    count=count+1;
    dist=(dist+abs(norm([px-px_old;py-py_old;pz-pz_old])));
    V_old=V;
    psi_old=psi;
    th_old=th;
    px_old=px;
    py_old=py;
    pz_old=pz;
end                     % the end of the "if" loop
end                     % the end of the "for" loop

AllSM= [time_SM Forces_SM Model_SM Pos_SM Vel_SM Acc_SM]; % update];
```

(end of function)

B. NCADEGUIDANCE.M

Description:

This MATLAB function develops the Directed Methods TS Guidance methodology to develop an optimal flight path for the air-launched interceptor

```
function path=NCADEGuidance(time,state,target,i)
% Written by LT John A. Lukacs IV, Naval Postgraduate School, June 2006
% Corrected by O.Yakimenko, August 2007, October 2009
% Adapted by Zheng Liang Lu,Singapore, May 2011
% This function takes in the state of the interceptor and target and
% generates an initial guess at the final conditions (position,
% orientation angles, range, and time to intercept) through a first-
order
% trajectory assumption and iterative process.  It then calls the
% fminsearch function using those initial guesses.  Finally, it plots
the
% returned optimal flight path and associated variables.
```

```
%% List of variables
% best      = vector of the variables in the optimal path returned from
%             the fminsearch function
% BC        = boundary conditions
% cost      = cost function value returned from NCADEGuidanceCost
function
% costs     = array of the value of the cost variables at each
iteration
% free      = variables that fminsearch can modify, specifically
%             [tau;thf;psif]
% init      = vector of initial estimates
% J         = vector of cost function variable values
% N         = length of the path vector (used for plotting)
% nmax      = maximum acceleration capability of the interceptor,
%             altitude dependent
% path      = returned time history of the optimal path
% psi       = initial interceptor heading angle
% psidot    = initial rate of change of interceptor heading angle
% psif      = final interceptor heading angle, calculated from final
%             conditions estimate
% psit      = target heading angle
% Py,Pz     = penalty functions on the y and z acceleration
% q         = variable, counting trajectory updates during intercept
% qq        = variable, counting the number of iterations during
optimization
% range     = estimate of distance between target and interceptor
% state     = state of the interceptor missile, sent from
NCADEGuidance.m
%             [px;py;pz;V;th;psi;Vdot;thetadot;psidot];
% states    = array of the values of all processes in NCADEGuidance.m
% target    = state of the rocket, sent from NCADEGuidance.m at i+w,
%             synchronizing times
% tau_f     = value of the virtual arc
% tgo       = time to go to intercept
% th        = initial interceptor flight path angle
% thdot     = initial rate of change of interceptor flight path angle
% thf       = final interceptor flight path angle
% tht       = target flight path angle
% tic..toc  = MATLAB function to track run time
% trys      = vector of optimal path and derivative values
% V         = initial interceptor velocity
% V_f       = final interceptor velocity
% Vave      = average interceptor velocity
% Vdot      = initial interceptor acceleration
% x0        = initial inteceptor position
% xd0       = initial interceptor velocity
% xdd0      = initial inteceptor acceleration
% xdf       = final inteceptor position
% xdt       = current target velocity
% xf        = final inteceptor acceleration
% xmult     = ratio value (used for plotting)
% xt        = current target position
% ymult     = ratio value (used for plotting)
% zmult     = ratio value (used for plotting)
```

```
global Re alphag path            % shared by SMFlight3, NCADEGuidance &
NCADETrajectory
global q states                  % shared by SMFlight3 & NCADEGuidance
global qq thdot psidot tgo costs trys          % ... by NCADEGuidance &
NCADETrajectory

%% Counting trajectory updates during intercept
q=q+1;

%% Initializing Interceptor (all states)
x0=state(1:3);
V=state(4);
th=state(5);
psi=state(6);
Vdot=state(7);
thdot=state(8);
psidot=state(9);
xd0  = [V*cos(th)*cos(psi);
        V*cos(th)*sin(psi);
        V*sin(th)];
xdd0 = [Vdot*cos(th)*cos(psi)-V*cos(th)*sin(psi)*psidot-
V*sin(th)*cos(psi)*thdot;
        Vdot*cos(th)*sin(psi)+V*cos(th)*cos(psi)*psidot-
V*sin(th)*sin(psi)*thdot;
        Vdot*sin(th)+V*cos(th)*thdot];

%% Initializing BM (up to the second-order derivatives)
xt  =target(1:3);
xdt =target(4:6);
xddt=target(7:9);

%% Estimating time-to-go
tgo1=100; delta=5;
vmaxhyp=2600;
while delta>1
    if tgo1>20
        V_f=vmaxhyp-10*(tgo1-20);
        Vave=(20*vmaxhyp/2+(tgo1-20)*(vmaxhyp+V_f)/2)/tgo1;
    else
        V_f=tgo1*vmaxhyp/20;
        Vave=V_f/2;
    end
    xf=xt+xdt*tgo1;
    tgo2=sqrt((xf(1)-x0(1))^2+(xf(2)-x0(2))^2+(xf(3)-x0(3))^2)...
         /(norm(xdt)+Vave);
    delta=abs(tgo2-tgo1);
    tgo1=(tgo1+tgo2)/2;
end
tgo=tgo1;

%% Initializing optimization
range=sqrt((xf(1)-x0(1))^2+((xf(2)-x0(2)))^2+((xf(3)-x0(3)))^2);
%fprintf('Trajectory update # %2.0f \n',q)
fprintf('\nSlant range to target:           %5.1fkm \n',range/10^3)
```

```
tau_f =0.00045*range-1000*(q-1);   % guess on tau_f
fprintf('Guess on the virtual arc length: %5.1f \n',tau_f)
fprintf('Guess on the time-to-go:         %5.1fkm \n',tgo)
    predicted_xdt=xddt*tgo;
tht =atan2(predicted_xdt(3),norm(predicted_xdt(1:2)));
psit=atan2(predicted_xdt(2),predicted_xdt(1));
thf   =0;%-tht;                  % guess on thf
psif  =psi;%psit+pi;             % guess on psif
free  =[tau_f;thf;psif;.1;1;1;-0.001;-0.001];
BC=[x0;xd0;xdd0;time;xt;xdt;xddt];

%% Searching for the minimum performance index
qq=0;   % counting iterations to converge
tic
options=optimset('MaxIter',100,'Tolfun',1,'TolX',1);
best = fminsearch(@(x) NCADETrajectory(x,BC),free,options);  %
Optimization
tcpu=toc;
fprintf('\nIt took %6.0f interations to converge\n',qq)
fprintf('Elapsed time is %6.1f seconds\n',tcpu),
fprintf('Combined performance index is %5.1f\n',costs(end,1))
fprintf('    including: tau_f=%5.1f, t2go=%5.1fs, ImpAngle=%4.1f
off,\n',...

costs(end,2),costs(end,3),180/pi*costs(end,4))
fprintf('                Pny=%5.1f and
Pnz=%5.1f\n',costs(end,5),costs(end,6))
fprintf('              Dtgo=%5.1f and
Altfine=%5.1f\n',costs(end,7),costs(end,8))

tau_f=best(1);
thf  =best(2);
psif =best(3);
[best(4);best(5);best(6);best(7);best(8)];

V_f=path(end,5);
xf=path(end,2:4);
fprintf(['Impact occurs at Altitude of %4.1fkm, Northing=%4.1fkm, and
'...
                              'Westing=%4.1fkm\n'],[xf(3)-Re xf(1)
xf(2)]/10^3)
fprintf('with interceptor''s speed of %4.1f km/s\n',V_f/10^3)

xdf=[V_f*cos(thf)*cos(psif);
     V_f*cos(thf)*sin(psif);
     V_f*sin(thf)];

%% Plotting results

xmult=20000/(norm(xd0)+norm(xdt));
ymult=20000/(norm(xd0)+norm(xdt));
zmult=20000/(norm(xd0)+norm(xdt));
nmax=40+(40-10)/(0-50000)*(path(:,4)-Re);
```

```
figure('Name','Bird-eye view')              % Bird-eye view
plot3(path(:,2)/10^3,path(:,3)/10^3,(path(:,4)-Re)/10^3,'-
.b','Linewidth',2)
hold on, grid
plot3(10^-3*[x0(1)-xmult*xd0(1);x0(1)+xmult*xd0(1)],...
      10^-3*[x0(2)-ymult*xd0(2);x0(2)+ymult*xd0(2)],...
      10^-3*[x0(3)-Re-zmult*xd0(3);x0(3)-
Re+zmult*xd0(3)],'c','Linewidth',2)
plot3(xf(1)/10^3,xf(2)/10^3,(xf(3)-Re)/10^3,'pr','Linewidth',2)
plot3(10^-3*[xf(1)-xmult*xdt(1);xf(1)+xmult*xdt(1)],...
      10^-3*[xf(2)-ymult*xdt(2);xf(2)+ymult*xdt(2)],...
      10^-3*[xf(3)-Re-zmult*xdt(3);xf(3)-
Re+zmult*xdt(3)],'r','Linewidth',2)
plot3(x0(1)/10^3,x0(2)/10^3,(x0(3)-Re)/10^3,'*b','Linewidth',5)
plot3((x0(1)+xmult*xd0(1))/10^3,(x0(2)+xmult*xd0(2))/10^3,...
                                (x0(3)-
Re+xmult*xd0(3))/10^3,'^c','linewidth',2)
plot3((xf(1)+xmult*xdt(1))/10^3,(xf(2)+xmult*xdt(2))/10^3,...
                                (xf(3)-
Re+xmult*xdt(3))/10^3,'^r','linewidth',2)
hl=legend('Intercept trajectory','Interceptor''s velocity vector',...
       'Impact point','BM''s velocity vector at
intercept','Location','Best');
set(hl,'FontSize',8);
xlabel('Northing (km)'), ylabel('Westing (km)'), zlabel('Altitude
(km)')
%title('Interception Geometery','Fontsize',10)
view(-102,8)
axis equal

%
figure('Name','Combined PI and Virtual arc')    % Performance index &
Virtual arc
subplot(211)
semilogy(costs(:,1)/costs(1,1),'h-.'), grid
xlabel('Iteration'), ylabel('Relative PI (PI_i/PI_1')
xlim([1 qq]), axis 'auto y' % ylim([0 2])
subplot(212)
plot(costs(:,2),'h-.'), grid
xlabel('Iteration'), ylabel('Length of virtual arc, \it\tau_f')
xlim([1 qq])

figure('Name','Delta Time-to-go and Altitude violation') % Dtgo &
Altitude fine
subplot(211)
plot(costs(:,7),'h-.'), grid
xlim([1 qq])
xlabel('Iteration'), ylabel('\Delta \itt_{go} \rm(s)')
subplot(212)
plot(costs(:,8)/10^3,'h-.'), grid
xlabel('Iteration'), ylabel('Alt. violation, (km)')
xlim([1 qq])

figure('Name','Impact angle and Time-to-go')     % Impact angle & Time-
to-go
```

```
subplot(211)
plot(real(180/pi*acos(costs(:,4))),'h-.'), grid
axis([1 qq 60 90])
xlabel('Iteration'), ylabel('Impact angle (^o)')
subplot(212)
plot(costs(:,3),'h-.'), grid
xlabel('Iteration'), ylabel('Time-to-go, \itt_{go} \rm(s)')
xlim([1 qq])

figure('Name','G-load factors')                    % G-load constraints
subplot(211)
plot(path(:,1),path(:,9),'-b.'), grid
hold on
plot(path(:,1),path(:,10),'--g.')
plot(path(:,1),nmax(:),'r','Linewidth',2)
plot(path(:,1),-nmax(:),'r','Linewidth',2)
hl=legend('n_y','n_z','Dynamic constraints',2);
set(hl,'FontSize',8);
xlabel('Time (s)'), ylabel('Load factor (g)')
subplot(212)
plot(costs(:,5)/10^7,'-b.','Linewidth',2), grid
hold on
plot(costs(:,6)/10^7,'--g.','Linewidth',2)
xlabel('Iteration'), ylabel('Relative penalty')
hl=legend('n_y penalty','n_z penalty','Location','Best');
set(hl,'FontSize',8);
xlim([1 qq])

figure('Name','Lambda and Tau profile')          % Lambda and tau
subplot(211)
plot(path(:,1),path(:,end),'.'), grid
xlabel('Time (s)'), ylabel('\it\lambda')
subplot(212)
plot(path(:,1),path(:,end-1),'.'), grid
xlabel('Time (s)'), ylabel('\it\tau')

figure('Name','Speed and Angle of attack profile') % SM speed and Angle
of attack
subplot(211)
plot(path(:,1),path(:,5),'.'); grid
xlabel('Time (s)'), ylabel('Speed, V (m/s)')
subplot(212)
plot(path(:,1),alphag,'.'), grid
xlabel('Time (s)'), ylabel('Angle of attack (^o)')

figure('Name','Euler angles profile')            % SM Euler angles
subplot(211)
plot(path(:,1),path(:,6)*180/pi,'.','Linewidth',2), grid
hold on
plot(path(end,1),thf*180/pi,'ro')
ylim([-30 90])
xlabel('Time (s)'), ylabel('\theta (^o)')
subplot(212)
plot(path(:,1),path(:,7)*180/pi,'g.','Linewidth',2), grid
hold on
```

```
plot(path(end,1),psif*180/pi,'ro')
ylim([-180 180])
xlabel('Time (s)'), ylabel('\psi (^o)')
%}

%% Creating results structure
states{q,1}=path;
states{q,2}=BC;
states{q,3}=free;
states{q,4}=best;
states{q,5}=costs;

return
```

(end of function)

C. NCADETRAJECTORY.M

Description:

This MATLAB function is sub-function of NCADEGuidance.m, which creates a 7^{th} order set of equations and evaluates that set at the boundary conditions supplied by the inputs to optimize the interceptor trajectory.

```
function cost=NCADETrajectory(free,BC)
% This function computes a candidate trajectory and associated cost
based on
% the vector of varied parameters "free" and boundary conditions "BC".

% This is a sub-funtion of the NCADEGuidance.m function's fminsearch.
% This function creates a 7th order set of equations and evaluates that
set at
% the boundary conditions supplied by the inputs. It then calculates
the time
% history of all the flight vehicle variables, including controls and
reactions,
% necessary to develop that flight path. A plot command set at the end
of this
% function will plot a chart of the iterations at the end of run if
desired.
% Finally, this function calculates the cost of the candidate
trajectory
% combining the value of the performance index and penalties. This cost
value is
% used to determine whether the proposed trajectory is optimal. (The
trajectory
% that returns the minimum value of the cost is the sub-optimal one.)

% O.Yakimenko, Naval Postgraduate School, November 2009
```

```
%% List of variables
% A,Ax,Axp,Axpp,Axppp    = cell matrices of coefficients of a candidate
reference
%                          trajectory and their derivatives wrt virtual
arc
% BC        = boundary conditions, specifically
[x0;xd0;xdd0;time;xt;xdt;xddt]
% Cx,Cxp,Cxpp,Cxppp      = coefficients of a candidate reference
trajectory and
%                          their derivatives wrt virtual arc
% dtau      = tau step value
% dtime     = time step value
% free      = variable parameters, specifically [tau;thf;psif]
% g         = gravitational force
% L         = lambda, virtual speed
% Lp        = first-order derivative of lambda wrt to virtual arc
% nmax      = maximum acceleration capability of the interceptor,
%             altitude dependent
% nX,nXp,nXpp,nXppp     = norm of reference trajectory and its
derivatives
% nx        = axial acceleration command, body frame x
% ny        = axial acceleration command, body frame y
% nz        = axial acceleration command, body frame z
% path      = returned time history of the optimal path, specifically
%             [time' X(1:3,:)' V' th' psi' nx' ny' nz' tau' L']
% psi       = interceptor heading angle
% psidot    = rate of change of interceptor heading angle
% qq        = variable counting the number of iterations
% t         = global current time
% tau       = virtual arc
% tau_f     = length of the virtual arc
% tgo       = time to go to intercept
% th        = interceptor flight path angle
% thdot     = rate of change of interceptor flight path angle
% thp       = first-order derivative of flight path angle wrt to
virtual arc
% time      = optimal path time history time=[0;tgo]
% trys      = collection of norms [X nX Xp nXp Xpp nXppp] (used for
plotting)
% V         = velocity
% Vdot      = acceleration
% Vp        = irst-order derivative of velocity wrt to virtual arc
% X,Xp,Xpp,Xppp         = reference trajectory and its derivatives wrt
tau
% x0,xf     = initial and final inteceptor position
% xd0,xdf   = initial and final interceptor velocity
% xdd0,xddf = initial and final inteceptor acceleration
% xp,xpp,xppp           = boundary conditions (at 0 and f) in the
virtual domain
% xt,xdt,xddt           = target position, velocity and acceleration at
time=0

global Re alphag path            % shared by SMFlight3, NCADEGuidance &
SMTrajectory
```

```
global Pos_BR Pos_SM Npol Npt kpolar weight90   % ... by SMFlight3 &
SMTrajectory
global qq thdot psidot tgo costs trys          % ... by NCADEGuidance &
SMTrajectory

%% Counting iterations to converge
qq=qq+1;
tgoold=tgo;

%% Assigning variables
tau_f=free(1);
thf  =free(2);
psif =free(3);
t=BC(10);  % current time in the SM frame to compute it's stage (thrust
and drug)

x0=BC(1:3);
xd0=BC(4:6);
xdd0=BC(7:9);
V(1)=norm(xd0);
Vdoti=norm(xdd0);

L(1)=1;%V(1);
Lpi=0;%Vdoti/L(1)

xt=BC(11:13);   % Target's coordinates at the moment of detection
xdt=BC(14:16);  % Target's velocities at the moment of detection
xddt=BC(17:19); % Target's accelerations at the moment of detection

V(Npt)=2600-1/3*(tgo-20);   % NACDE velocity estimate at impact
Vdotf=-0.3;%-5.9578;        % NCADE acceleration estimate at impact

L(Npt)=1;%V(Npt);
Lpf=0;%Vdotf/L(Npt);

xf=xt+xdt*tgo+0.5*xddt*tgo^2;
xdf=[V(Npt)*cos(thf)*cos(psif);
     V(Npt)*cos(thf)*sin(psif);
     V(Npt)*sin(thf)];

thdotf=free(7); psidotf=free(8);
xddf=[Vdotf*cos(thf)*cos(psif)-V(Npt)*cos(thf)*sin(psif)*psidotf-...
      V(Npt)*sin(thf)*cos(psif)*thdotf;
      Vdotf*cos(thf)*sin(psif)+V(Npt)*cos(thf)*cos(psif)*psidotf-...
      V(Npt)*sin(thf)*sin(psif)*thdotf;
      Vdotf*sin(thf)+V(Npt)*cos(thf)*thdotf];

%% Converting boundary conditions ito the virtual domain
xp0=xd0/L(1);
xpp0=(xdd0-xd0*Lpi)/L(1)^2;
xppp0=[free(4);free(5);free(6)];

xpf=xdf/L(Npt);
```

```
xppf=(xddf-xdf*Lpf)/L(Npt)^2;
xpppf=[0;0;0];

%% Calculating polynomials' coefficients and reference trajectories
    dtau =tau_f/(Npt-1);
    tau  =linspace(0,tau_f,Npt);
for i=1:3
A{i}=[x0(i);
      xp0(i);
      xpp0(i);
      xppp0(i);
      (-16*xppp0(i)-4*xpppf(i))/tau_f+(-
120*xpp0(i)+60*xppf(i))/tau_f^2+...
                  (-360*xpf(i)-480*xp0(i))/tau_f^3+(840*xf(i)-
840*x0(i))/tau_f^4;
      (60*xppp0(i)+30*xpppf(i))/tau_f^2+(600*xpp0(i)-
420*xppf(i))/tau_f^3+...
                (2340*xpf(i)+2700*xp0(i))/tau_f^4+(5040*x0(i)-
5040*xf(i))/tau_f^5;
      (-80*xppp0(i)-60*xpppf(i))/tau_f^3+(780*xppf(i)-
900*xpp0(i))/tau_f^4+...
             (-4080*xpf(i)-4320*xp0(i))/tau_f^5+(-
8400*x0(i)+8400*xf(i))/tau_f^6;
      (35*xppp0(i)+35*xpppf(i))/tau_f^4+(420*xpp0(i)-
420*xppf(i))/tau_f^5+...
               (2100*xpf(i)+2100*xp0(i))/tau_f^6+(4200*x0(i)-
4200*xf(i))/tau_f^7];
    Ax{i}   =diag([ 1,   1,   1/2, 1/6,  1/24, 1/60, 1/120, 1/210
])*A{i};
    Axp{i}  =diag([ 0,   1,   1,   1/2,  1/6,  1/12, 1/20,  1/30
])*A{i};
    Axpp{i} =diag([ 0,   0,   1,   1,    1/2,  1/3,  1/4,   1/5
])*A{i};
    Axppp{i}=diag([ 0,   0,   0,   1,    1,    1,    1,     1  ])*A{i};
    Cx(i,:)   =Ax{i}([Npol:-1:1]);
    Cxp(i,:)  =Axp{i}([Npol:-1:2]);
    Cxpp(i,:) =Axpp{i}([Npol:-1:3]);
    Cxppp(i,:)=Axppp{i}([Npol:-1:4]);
    X(i,:)   =polyval(Cx(i,:),tau);
    Xp(i,:)  =polyval(Cxp(i,:),tau);
    Xpp(i,:) =polyval(Cxpp(i,:),tau);
    Xppp(i,:)=polyval(Cxppp(i,:),tau);
end

%% Computing the states
xp12=Xp(1,:).^2+Xp(2,:).^2;
th  =atan2(Xp(3,:),sqrt(xp12));
psi =atan2(Xp(2,:),Xp(1,:));
thp =(Xpp(3,:).*xp12-
Xp(3,:).*(Xp(1,:).*Xpp(1,:)+Xp(2,:).*Xpp(2,:))./...

sqrt(xp12)./(xp12+Xp(3,:).^2);
psip =(Xp(1,:).*Xpp(2,:)-Xpp(1,:).*Xp(2,:))./xp12;

time(1)=t;
```

```
g      = 3.986004418e14/norm(X(1:3,1))^2;
nx(1)  = Vdoti/g+sin(th(1));
ny(1)  = V(1)/g*cos(th(1))*psidot;
nz(1)  = V(1)/g*thdot+cos(th(1));

[ro,press,temp]=STatmos(norm(X(:,1))-Re);
[m_i,Sref]=NCADEParams1(time(1));
alphag(1)=180/pi*sqrt(ny(1)^2+nz(1)^2)*m_i*g/(ro*V(1)^2*Sref/2)/13;

for j=2:Npt;
    g=3.986004418e14/norm(X(1:3,j))^2;
        if norm(X(:,j))-Re<86000, [ro,press,temp]=STatmos(norm(X(:,j))-
Re);
        else                      [ro,press,temp]=STatmos(86000);
end
    MV=V(j-1)/sqrt(1.402*287.053*temp);
    CDtable=NCADEDrag(MV);
        if      time(j-1)<=6,   Thrust=206000; CD0=CDtable(1);

        elseif  time(j-1)<20    Thrust=95300;  CD0=CDtable(1);

        else                    Thrust=0;      CD0=CDtable(2);
end
    [m_i,Sref]=NCADEParams1(time(j-1));
            CL=(2*sqrt(ny(j-1)^2+nz(j-1)^2)*m_i*g)/(ro*V(j-1)^2*Sref);
            CD = CD0 + kpolar*CL^2;
    Drag=ro*V(j-1)^2*CD*Sref/2;
    nx(j)=(Thrust-Drag)/m_i/g;

    V(j)=V(j-1)+g*(nx(j-1)-sin(th(j-1)))/L(j-1)*dtau;

    ddist=sqrt((X(1,j)-X(1,j-1))^2+(X(2,j)-X(2,j-1))^2+(X(3,j)-X(3,j-
1))^2);
    dtime=2*ddist/(V(j)+V(j-1));
    L(j)=dtau/dtime;

    ny(j)=V(j)/g*cos(th(j))*psip(j)*L(j);
    nz(j)=V(j)/g*thp(j)*L(j)+cos(th(j));
    alphag(j)=180/pi*sqrt(ny(j)^2+nz(j)^2)*m_i*g/(ro*V(j)^2*Sref/2)/13;

    time(j)=time(j-1)+dtime;
end

tgo=time(end)-time(1);

for i=1:Npt
    nX(i)=norm(X(1:3,i));
    nXp(i)=norm(Xp(1:3,i));
    nXpp(i)=norm(Xpp(1:3,i));
end

trys=[X' nX' Xp' nXp' Xpp' nXpp'];
path=[time' X(1:3,:)' V' th' psi' nx' ny' nz' tau' L'];
```

```
%% Computing cost and penalties
V_f=V(end);
xdt=BC(14:16)+BC(17:19)*tgo;
xdf=[V_f*cos(thf)*cos(psif);
     V_f*cos(thf)*sin(psif);
     V_f*sin(thf)];

nmax=40+(40-10)/(0-50000)*(X(3,:)-Re);
if nmax<1, nmax=1; end

J=[tgo;
   abs(dot(xdf,xdt))/norm(xdf)/norm(xdt)];
Py=max([0,abs(ny)-nmax]);
Pz=max([0,abs(nz)-nmax]);
Dtgo=tgoold-tgo;
Altfine=max([0,X(3,end)-Re-60000]).^2+min([0,X(3,end)-Re-9000]).^2;
cost=norm([1,weight90]*J)+10*(Py^2+Pz^2)+100*Dtgo^2+0.1*Altfine;
costs(qq,:)=[cost;tau_f;J;Py;Pz;Dtgo;Altfine];

%% Animating iterations
%{
figure(100),% set(gcf,'Color','w');
subplot(3,6,[1 2 7 8])
plot3(Pos_BR(1:300,1)/10^3,Pos_BR(1:300,2)/10^3,(Pos_BR(1:300,3)-
Re)/10^3,...
                                    '-.r','LineWidth',2)
hold on
plot3(Pos_BR(140,1)/10^3,Pos_BR(140,2)/10^3,(Pos_BR(140,3)-Re)/10^3,...
                                    '^g','LineWidth',2)
plot3(Pos_SM(1:19,1)/10^3,Pos_SM(1:19,2)/10^3,(Pos_SM(1:19,3)-
Re)/10^3,...
                                    '.k','LineWidth',2)

plot3(X(1,:)/10^3,X(2,:)/10^3,(X(3,:)-Re)/10^3,'Linewidth',3); grid on
plot3(xf(1)/10^3,xf(2)/10^3,(xf(3)-Re)/10^3,'pk','MarkerSize',11)
plot3(X(1,end)/10^3,X(2,end)/10^3,(X(3,end)-
Re)/10^3,'pr','MarkerSize',9);
view(-102,8)
hl=legend('BM trajectory','BM detection','Unguided ascend',...
       'Guided flight','Predicted intercept point','Actual intercept
point',...
       'Location','Best'); set(hl,'FontSize',7)
hold off
axis([0 1.1e5 -1e4 1.1e5 0 7e4]/10^3)
xlabel('Northing, x (km)'),ylabel('Westing, y (km)'),zlabel('Altitude
(km)')
        subplot(3,6,[13 14])
        plot(path(:,1),path(:,9),'--b','Linewidth',2)
        hold on; grid on
        plot(path(:,1),path(:,10),'-.g','Linewidth',2)
        plot(path(:,1),nmax(:),'r','Linewidth',2)
        plot(path(:,1),-nmax(:),'r','Linewidth',2)
        axis([10 time(Npt) -45 45]);
```

```
        xlabel('Time (s)'), ylabel('Load Factor (g)')
        leg2=legend('n_y','n_z','Dynamic
constraints','Location','Best');
        set(leg2,'FontSize',7)
        hold off
      subplot(3,6,3)
      plot(time,X(1,:)/10^3,'Linewidth',2); grid on
      axis([10 time(Npt) 1e4/10^3 1.1e5/10^3])
      title('x_1 (km)')
    subplot(3,6,9)
    plot(time,X(2,:)/10^3,'Linewidth',2); grid on
    axis([10 time(Npt) -1e4/10^3 1.1e5/10^3])
    title('x_2 (km)')
  subplot(3,6,15)
  plot(time,(X(3,:)-Re)/10^3,'Linewidth',2); grid on
  axis([10 time(Npt) 0 7e4/10^3])
  title('x_3 (km)'), xlabel('Time (s)')
subplot(3,6,4)
plot(time,Xp(1,:)/10^3,'Linewidth',2); grid on
axis([10 time(Npt) -45 45]); axis 'auto y'
title('x_1'' /10^3')
  subplot(3,6,10)
  plot(time,Xp(2,:)/10^3,'Linewidth',2); grid on
  axis([10 time(Npt) -45 45]); axis 'auto y'
  title('x_2'' /10^3')
    subplot(3,6,16)
    plot(time,Xp(3,:)/10^3,'Linewidth',2); grid on
    axis([10 time(Npt) -45 45]); axis 'auto y'
    title('x_3'' /10^3'), xlabel('Time (s)')
      subplot(3,6,5)
      plot(time,Xpp(1,:)/10^2,'Linewidth',2); grid on
      axis([10 time(Npt) -45 45]); axis 'auto y'
      title('x_1'''' /10^2')
        subplot(3,6,11)
        plot(time,Xpp(2,:)/10^2,'Linewidth',2); grid on
        axis([10 time(Npt) -45 45]); axis 'auto y'
        title('x_2'''' /10^2')
      subplot(3,6,17)
      plot(time,Xpp(3,:)/10^2,'Linewidth',2); grid on
      axis([10 time(Npt) -45 45]); axis 'auto y'
      title('x_3'''' /10^2'), xlabel('Time (s)')
    subplot(3,6,6)
    hold on
    plot(qq,time(Npt)-time(1),'+c','Linewidth',1); grid on
    hold off
    axis([1 200 40 60]); axis 'auto y'
    title('Time-to-go (s)')
  subplot(3,6,[12 18])
  hold on
  plot(qq,cost,'+m','Linewidth',1); grid on
  axis([1 200 0 100]); axis 'auto y'
  hold off
  title('Performance Index'), xlabel('Iteration')
%}
return
```

(end of function)

D. NCADEPARAMS1.M

Description:

This MATLAB function calculates the J-Matrix and the Mass of the 2-stage Air-launched NCADE Interceptor.

```
function [SM_mass,Sref]=NCADEParams1(t)
% Written originally by LT John A. Lukacs IV, Naval Postgraduate
School, June 2006
% Modified by Zheng Liang Lu, Singapore MOD, May 2011
% This function calculates the J Matrix and Mass of the air-launched
NCADE interceptor,
% assuming a cruciform rocket in two stages to intercept.  This
% function also returns the reference (base) diameter of the missile.
% The NCADE is a derivative of the AIM-120 AMRAAM
% Specific NCADE data is classified or proprietary to the missile OEM
% hence, assumptions must be made about the NCADE's physical and
operational parameters
% It is assumed that the NCADE dimensions and mass are similar to the
% AIM-120 and its trajectory similar to the SM3 and SM6 ABMs

%% Notes:
% The first stage last 6 seconds, the second stage lasts an additional
% 10 seconds.  Stage 1 (booster) seperates upon completion.  Stage 2
% does not separate after completion

%% List of variables
% dia           = reference diameter, base diameter (in m)
% l             = length, varies by component (in m)
% p_SM_st1_fuel = density of stage 1 rocket fuel
% p_SM_st2_fuel = density of stage 2 rocket fuel
% p_SMstr       = density of structural material
% r             = radius, varies by component
% ro            = outer radius, varies by component
% ri            = inner radius, varies by component
% SM_mass       = total rocket mass
% SM_nose       = total mass of nosecone section
% SM_st1_fcr    = consumption rate of stage 1 fuel
% SM_st1_fuel   = remaining stage 1 fuel based on time and
%                 consumption rate
% SM_st1_str    = total mass of stage 1 structural material
% SM_st1_tfm    = total mass of stage 1 fuel
% SM_st2_fcr    = consumption rate of stage 2 fuel
% SM_st2_fuel   = remaining stage 2 fuel based on time and
%                 consumption rate
% SM_st2_str    = total mass of stage 2 structural material
% SM_st2_tfm    = total mass of stage 2 fuel
% t             = time
```

```
% th            = structural thickness
% V_body        = volume of body structural material
% V_nose_str    = volume of nosecone structural material
% V_nose_str0   = volume of nosecone structural material,
%                 intermediate value
% V_nose_str1   = volume of nosecone structural material,
%                 intermediate value
% V_st1_fuel    = volume of stage 1 fuel
% V_st1_str     = volume of stage 1 structural material
% V_st2_fuel    = volume of stage 2 fuel
% V_st2_str     = volume of stage 2 structural material

%% Structural components
    p_SMstr = 4225;
    th = .0208;

    %overall length of AIM-120 is 3.66m, wingspan is 0.5258m IAW USAF
Fact
    %Sheet
    % the length of the NCADE 1st stage is 1.6529m and the 2nd stage
    % (interceptor-engine and warhead) is 2.0071m.
    % Mass of nose cone
    l = 0.5608;
    r = 0.1778/2;
    V_nose_str0 = pi*(l*((r^2+l^2)/(2*r))^2-l^3/3-(((r^2+l^2)/...
          (2*r))-r)*((r^2+l^2)/(2*r))^2*asin(l/((r^2+l^2)/(2*r))));
    l = 0.5608-th;
    r = 0.1778/2-th;
    V_nose_str1 = pi*(l*((r^2+l^2)/(2*r))^2-l^3/3-(((r^2+l^2)/...
          (2*r))-r)*((r^2+l^2)/(2*r))^2*asin(l/((r^2+l^2)/(2*r))));
    V_nose_str = V_nose_str0-V_nose_str1+pi*r^2*th;
    SM_nose = 1.3*V_nose_str*p_SMstr;

    % Mass of Body/Warhead Section
    l = 0.5608;
    ro = 0.1778/2;
    ri = 0.1778/2-th;
    V_body = l*pi*(ro^2-ri^2);
    SM_body = V_body*p_SMstr+115;

    % Mass of Stage 1 (NCADE Booster)
    l = 1.6529;
    ro = 0.1778/2;
    ri = 0.1778/2-th;
    V_st1_str = l*pi*(ro^2-ri^2)+2*pi*ro^2*th;
    SM_st1_str = V_st1_str*p_SMstr;

    % Mass of Stage 2 (Engine and Warhead Section)
    l = 2.0071-2*th;
    ro = 0.1778/2;
    ri = 0.1778/2-th;
    V_st2_str = l*pi*(ro^2-ri^2)+2*pi*ro^2*th;
    SM_st2_str = V_st2_str*p_SMstr;
```

```
%% Fuel Components
    % Stage 1 Solid Fuel is HTPB/AP/Al
    l = 1.6529;
    ri = 0.1778/2-th;
    V_st1_fuel = 0.80*l*pi*ri^2;
    p_SM_st1_fuel = 1860;
    SM_st1_tfm = 468;
    SM_st1_fcr = 468/6;
    % Stage 2 Solid Fuel is TP-H1205/6
    l = 2.0071-0.5608;
    ri = 0.1778/2-th;
    V_st2_fuel = 0.60*l*pi*ri^2;
    SM_st2_tfm = 360;
    p_SM_st2_fuel = SM_st2_tfm/V_st2_fuel;
    SM_st2_fcr = 360/15;

if t<6
    %% Stage 1 - Stage 1 Fuel is consumed, Stage 2 Fuel is not used.
    SM_st1_fuel = SM_st1_tfm - SM_st1_fcr * t;
    SM_mass = SM_nose+SM_body+SM_st2_str+SM_st2_tfm+SM_st1_str+...
          SM_st1_fuel;
    dia = 0.1778;

elseif t<21
    %% Stage 2 - Stage 1 has seperated, Stage 2 Fuel is consumed.
    SM_st2_fuel = SM_st2_tfm - SM_st2_fcr * (t-6);
    SM_mass = SM_nose+SM_body+SM_st2_str+SM_st2_fuel;
    dia = 0.1778;

else
    %% Stage 3 - The unpowered nosecone and Stage 2 remains.
    SM_mass = SM_nose+SM_body+SM_st2_str;
    dia = 0.1778;

end

Sref=pi*dia^2/4;

return
```

(end of function)

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Yeo Tat Soon, Director TDSI
   Temasek Defence Systems Institute (TDSI)
   Singapore, Republic of Singapore

4. Ms Tan Lai Poh, Senior Manager TDSI
   Temasek Defence Systems Institute (TDSI)
   Singapore, Republic of Singapore

5. Lu Zheng Liang
   Singapore Technologies Aerospace
   Singapore, Republic of Singapore